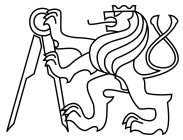
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Podpora vývoje aplikací pro čipové karty Java Card
Student:	Valeriya Pak
Vedoucí:	Ing. Jiří Buček
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Prostudujte volně dostupné vývojové nástroje pro vývoj appletů na platformě Java Card. Proveďte aktualizaci existujících vývojových modulů z předchozích prací [1, 2] na aktuální verzi Netbeans. Doplňte potřebné vybavení (zásuvné moduly, konfiguraci), aby bylo možno vyvíjet applety pro Java karty, které podporují Secure Channel Protocol 2 (SCP02), případně další protokoly po dohodě s vedoucím práce. Výsledek práce otestujte na kartě typu NXP J3A.

Seznam odborné literatury

[1] Petr Vlášek: Demonstrace šifrování na platformě Java Card, diplomová práce, ČVUT v Praze, 2008.
[2] Filip Munzar: Podpora vývoje aplikací pro čipové karty Java Card, bakalářská práce, ČVUT v Praze, 2014.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 3. března 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Podpora vývoje aplikací pro čipové karty Java Card

Valeriya Pak

Vedoucí práce: Ing. Jiří Buček

30. června 2017

Poděkování

Děkuji Ing. Jiřímu Bučkovi za rady a trpělivost.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 30. června 2017 .....

Odkaz na tuto práci

Pak, Valeriya. *Podpora vývoje aplikací pro čipové karty Java Card.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Probírají se především mechanismy zabezpečení komunikace se smart karty pomocí SCP01 a SCP02 podle specifikace GlobalPlarform. Práce popisuje mezinárodní standardy, technologií čipových karet na platformě Java Card. V rámci této práce sada modulů Java Card Manager Suite pro vyvovjove prostředí NetBeans IDE byla rozšířena o podporu protokolu SCP02.

Klíčová slova Čipová karta, Java Card, Java applet, GlobalPlatform, NetBeans IDE

Abstract

This thesis describes secure communication with Smart Card based on GlobalPlatform specification v2.2. It also describes other standards related Smart Cards, general principles of the Java Card technology and NetBeans Platform. As a part of this thesis the Java Card Manager Suite was updated to support SCP02.

Keywords Smart Card, Java Card, Java applet, GlobalPlatform, NetBeans IDE

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Chytré karty jsou od svého vzniku velice aktuální a postupně se zavádí ve stále více odvětví lidské činnosti. Zpracování dat v kartě zaručují aplikace, které jsou schopné běžet na zařízení s omezeným výpočetním výkonem a malou paměťovou kapacitou

Platforma Java Card umožňuje vývoji a běh applikaci, tzv. 'appletu', na chytrých kartách. Pro snadnější vývoj appletu a jejich nasazení Ing. Petr Vlášek v rámci své diplomové práce „Demonstrace šifrování na platformě Java Card"[1] vytvořil sadu modulu Java Card Manager Suitě pro vývojové prosredi NetBeans IDE verze 6.0. Bc. Filip Munzar v práci „Podpora vývoje aplikací pro čipové karty Java Card"[2] přepsal tyto moduly, aby je bylo možné používat v NetBeans IDE ve verzi 8.0.

Cílem práce je aktualizovat sadu modulu Java Card Manager Suite pro Netbeans IDE ve verzi 8.2: otestovat a případně je přepsat. Dalším úkolem je rozšíření funkčnosti o podporu protkolu zabezpečeného kanálu SCP02.

Textová část práce se zaměřuje na potřebné teoretické základy. Obsahuje informaci o technologii Smart Card. Probírá se platforma Java Card, NetBeans a alternativní vývojové nástroje. Jelikož implementace podpory zabezpečeného komunikačního protokolu SCP02 je primární cíli této práce jsou popsané průběh komunikace s chytrými kartami a její zabezpečení v rámci specifikace GlobalPlatform.

Praktická část se věnuje úpravám v Java Card Manager Suitě a implementaci podpory SCP02.

Kapitola 1

Teoreticka cast

Obsahem této kapitoly je přehled použitých nástrojů a technologií. Jsou popsané Smart Card, Java Card, GlobalPlatform, platforma NetBeans a vývojové prostředí NetBeans IDE. Také jsou probrané alternativní vývojové nástroje.

1.1 Smart Card

Smart Card je čipová karta s integrovaným obvodem (proto bývají označovány jako ICC karty), který je zodpovědný za zpracování veškerých dat. Samotný obvod se skládá z mikroprocesorů s vstupním a výstupním rozhraním a pamětí.

Ze softwarového pohledu se na smart kartě nachází: operační systém, aplikace a v kartách s interpretem jsou běhové prostředí[1] a zavaděč[2].

1.1.1 Klasifikace karet

Karty lze klassifikovat podle několika kritérií. Zvolený způsob zohledňuje nejvýznamnější atributy, které primárně ovlivňují možný způsob použití, resp. aplikaci karet.

Podle způsobu komunikace karty se dělí na 3 typy: kontaktní, bezkontaktní a s dvojitým rozhraním.

Kontaktní smart karta obsahuje výše uvedené součástky a kontaktní oblast s 6 nebo 8 kontakty, pomocí kterou se spojí s čtečkou.

Bezkontaktní smart karta obsahuje rovněž výše popsané části, RFID čip a vestavěnou anténu, umístěnou po celém obvodu karty, kterou zprostředkovává elektromagnetické vlny.

[1] Komponenta, obsahující interpret. Stará se o běh aplikací na čipové kartě a také o knidhovny těchto aplikací

[2] Obstarává nahrávání a odstraňování aplikací z karty

Smart karty s dvojitým rozhraním se využívají buď jako kontaktní nebo bezkontaktní podle potřeby.

Z hlediska vnitni architektury rozlišujeme karty paměťové a mikroprocesorové.

Paměťové jsou osazené EPROM a EEPROM paměti a obvodem. Obvykle neumí kryptovat a poskytují základní zabezpečení. Jsou levnější než mikroprocesorové a vzhledem ke své jednoduchosti se využívají pro jednoduché aplikace.

Mikroprocesorové navíc obsahují volatilni paměť a mikroprocessor. EEPROM paměť poskytuje souborový systém, volatilni RAM paměť se používá pro ukládání dočasných dát mikroprocesoru. Také v kartě může být implementován kryptografický koprocesor.

Též lze rozlišovat karty podle využitého operačního systému, tzv. COS.

Dedikovaný je určený výhradně pro obsluhu jedné aplikace.

Multiaplikační dovoluje instalaci a běh několika aplikace.

COS spravuje souborový systém, přenos dat mezi kartou a čtečkou, spouštění aplikace a poskytuje API.

1.1.2 Standardy a specifikace

Jelikož čipová karta je jedna z komponent komplexního systému, nejen její samotné vlastnosti musejí být definované, ale také způsoby komunikace nebo rozhraní. Hlavním účelem standardů a specifikace je zajištění, že samotné karty, čtecí zařízení a aplikace od různých výrobců budou schopné spolupracovat.

1.2 Java Card

Platforma Java Card je technologie z rodiny Java, která umožňuje vývoj a poskytuje běh aplikací nazývaných applety. Je multiplatformni, zaručuje bezpečný běh appletu, a je kompatibilní s existujícími mezinárodními standardy: ISO, EMV a GlobalPlatform1.1.2.

Od Java Card 3.0 je technologie Java Card k dispozici ve dvou edicích. První z nich je Classic edition a druhou pak Connected edition. Classic edition je nástupce specifikace Java Card 2.2.2, zaměřená na současné použití čipových karet a založená na ISO/IEC 7816 a ISO/IEC 14443. Java Card Connected edition přidává podporu modelu webové aplikace. Tato práce je zaměřena na Classic edition.

Technologie Java Card má tři stavebně kameny:

Tabulka 1.1: Přehled standardů chytrých karet

Název	**Popis**
ISO/IEC 7816	Je vyvynut Mezinárodní organizace pro normalizaci a je nejdůležitější standard, který popisuje charakeristiky čipove karty. 1. část definuje fyzikální vlastnosti karty, obsahem 2. kapitoly jsou rozměry a umístění kontaktního modulu, 3. pospisuje konkrétní význam a použití jednotlivých vývodů konektorů a přenosové protokoly, 4. se věnuje aplikačnímu protokolu, struktuře uložených dat a metodám přístupu k nim. Části 5. – 15. jsou dodatky a vysvětlivky k ISO 7816-3 a ISO 7816-4.
ISO/IEC 14443	Mezinárodní norma popiujici bezkontaktní karty a přenosové protokoly pro komunikaci s nimi.
EMV	Specifikace platebních karet a transakčních terminálu. Vychází z normy ISO/IEC 7816 a lze považovat za rozšíření zaměřené na finanční operace.
PC/SC (Personal Computer/Smart Card)	Popisuje integraci chytrých karet do počítačových systému. Stanovuje API uživatelských aplikací a driverů čtecích zařízeních.
GlobalPlatform	Viz kapitola 1.4

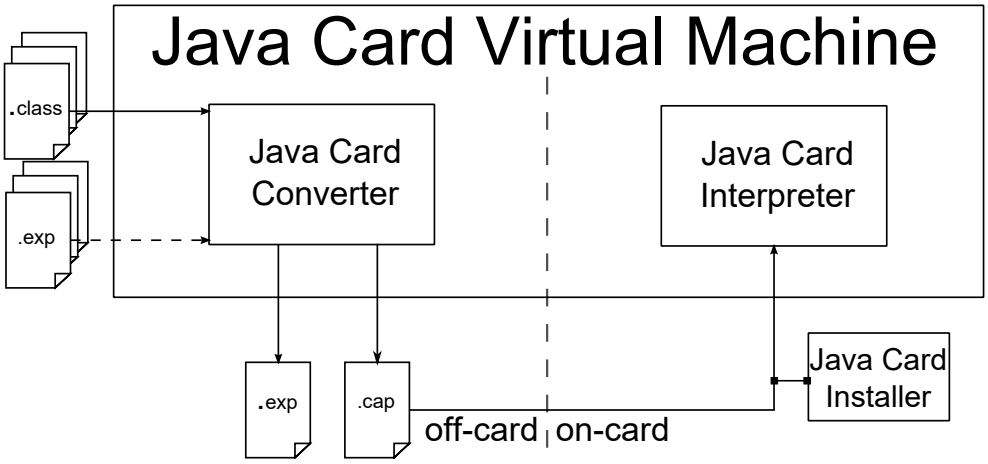
- Java Card Virtual Machine definuje podmnožinu programovacího jazyka Java a virtuální stroj pro Java Card.
- Java Card Runtime Environment určuje chování při kompilaci a spuštění programu Java – popisuje správu paměťové oblasti, zacházení s appletem atd.
- Java Card API specifikuje třídy a jejich metody pro vývoj appletů.

1.2.1 Virtuální stroj Java Card a jeho omezení

Na rozdíl od klasické Java Virtual Machine (JVM) Java Card Virtual Machine (JVCM) tvoří dvě části: konvertor (JavaCard Converter) a interpret (JavaCard Interpreter).

Konvertor je umístěn mimo kartu (off-card) a zpracovává `.class` soubory tříd, které tvoří balík. Výsledkem je `.cap` soubor, který je potom nahrán na čipovou kartu. Spolu s `.cap` souborem konvertor vytváří také `.exp` soubory, které obsahují jenom popis veřejných tříd balíku. Konverter může brát jako vstup `.exp` soubory v případě konvertace balička, který importuje třídy z jiného. Také konverter kontroluje správnost byte kódu a provádí jeho optimizace, detekuje nedodržení omezení definujících jazyk Java Card[3].

Interpret je on-card součástkou, tzn. že pracuje přímo v kartě: provádí instrukce bajtkodu, zajišťuje bezpečnost v době běhu appletu a spravuje paměť. Právě interpret poskytuje jednu z největších výhod platformy Java Card – nezávislost na hardwaru. Interpret nezavádí sám `.cap` soubory, a pouze provádí kód, který se v nich nachází. Zavádění má na starosti instalátor, který není součástí JVCM. Důvodem rozdělení je zachování malou velikostí jak instalátorů tak JVCM[4].



Obrázek 1.1: Virtuální počítač JavaCard

Typické zařízení s omezeným výpočetním výkonem má kolem 1.2K RAM, 16K non-volatilni paměti, průměrně 40K ROM a na implementace některých prvků klasické Javy by nestačilo místo. Proto specifikace JCVM definuje omezení podmnožiny jazyků Java Card.

Je nutné zmínit, že virtuální stroj Java Card Connected edition obsahuje výrazná rozšíření oproti klasické platformě: je schopen pracovat s více vlákny a tedy i podporovat garbage collector, obsahuje širší API, podporuje servlety [3] a HTML rozhraní.

1.2.2 Java Card applety

Applet je aplikace na platformě Java Card a musí rozšiřovat abstraktní třídu `javacard.framework.Applet`, jejíž obsahem jsou definice metod zodpovědných za komunikace mezi appletem a běhovým prostředí Java Card: `install, select, deselect, process`.

Statická funkce `install` slouží k vytvoření instance třídy applet a jeho zaregistrování, musí přímo nebo nepřímo volat metodu `register`, která daný applet zaregistruje a přiřadí mu hexadecimální identifikátor AID, kterým bude daný applet vybírán. Metoda `select` je zavolána, když je daný applet vybrán

[3] Nástroj pro tvorbu webových aplikací

Tabulka 1.2: Podmnožina Java Card

Nepodporované	**Podporované**
Funkcionalita	
Dynamické nahrávání tříd	Balíčky
Security Manager	Dynamické vytváření objektů
Finalizace	Virtuální metody
Vlákna	Rozhraní
Klonování	Výjimky
Typově bezpečné výčtové typy	Generické datové typy
Variabilní počet argumentů	Statický import
Klíčová slova	
	`abstract`
	`boolean`
	`break`
	`case`
	`catch`
	`class`
	`continue`
	`default`
	`do`
`native`	`else`
`strictfp`	`extends`
`synchronized`	`final`
`enum`	`finally`
`transient`	`for`
`assert`	`goto`
`volatile`	`if`
	`implements`
	`import`
	`instanceof`
	`int`
	`interface`
	`new`
	`package`
	`private`
Typy	
`char`	`boolean`
`double`	`byte`
`float`	`short`
`long`	`int`
Vícerozměrná pole	Objekty
	Jednorozmerná pole

přes APDU příkaz SELECT. Applet může své vybrání odmítnout pomocí návratové hodnoty false, nebo vyhozením výjimky. V případě, že metoda `select` vrátí true, APDU příkazy jsou dále zpracováne metodou `process`. Metoda `deselect` appletu je zavolána v případě, že je daný applet vybrán a je zavolán příkaz SELECT obsahující AID jiného appletu. Metoda dává možnost appletu provést operace nutné pro jeho správné ukončení [5].

implicitně applety jsou vzájemně nezávislé a nemůžou mezi sebou komunikovat, ale pokud applet implementuje rozhraní `javacard.framework.Shareble`, pak může sdílet některé své metody. Přecházet mez mezi applety můžou jenom primitvni typy a reference na APDU bufer[4]. Stejně jako Java dovoluje vytváření balíčků tříd a rozhraní, Java Card umožňuje vytváření balíčků ze souvisejících appletů.

1.2.2.1 Alternativy Java Card

Java Card není jediná vývojová platfroma pro čipové karty. Obsahem této sekce jsou stručné popisy alternativních platforem.

MULTOS je otevřený multiaplikační systém. Certifikován na nejvyšší úroveň zabezpečení dle standardu ITSEC E6[5]. Používá se převážně v oblasti řízení přístupu, bankovnictví a platebních systémech, elektronických pásech nebo autentizacnich systémech. Aplikace můžou být vytvořené buď pomocí nového jazyka nazvaného MEL, nebo C a Java pro které je poskytován překladač na jazyk MEL. Také používá virtuální stroj a díky tomu vývoj aplikací není závislý na nějakém konkrétním typu karty[6].

.NET Card je softwarová platforma a byla firmou HiveMinded, která ji implementovala do smart karet jako .NET Card framework, který vznikl pod patronatem firmy Microsoft. Aplikace pro technologii .NET Card lze programovat pomocí různých programovacích jazyků a také je možné použít v kódu jedné aplikace více programovacích jazyků zároveň, protože je daný kód aplikace následně překompilován do .NET kódu. Stejně jako technologie Java Card je to multiplikační prostředí a obsahuje virtuální stroj.[7].

1.3 Komunikace s chytrými kartami

Tato část se věnuje komunikaci mezi čtečkou a kartou. Popisuje průběh komunikace a vysvětluje některé pojmy, které pak jsou použité v 1.4.

Celá komunikace je zahájena vložením čipové karty do čtečky a v tomto momentě je karta resetována a první informací, kterou pošle čtečce nazpět,

[4] Misto v kartě, kde obvykle se rozmísťuje přijaty APDU příkaz a vytváří se odpověď na nej.

[5] The Information Technology Security Evaluation Criteria – sada kritérií pro hodnocení pocitacove bezpecnosti

je ATR. Samotná komunikace je vždy založena na principu master-slave, kde čtečka je v roli master a posílá příkaz do Smart karty, která je zpracovává a posílá odpověď čtečce. Komunikace probíhá pomocí přenosového protokolu, nejpouziavnejsi jsou T=0 a T=1. Do přenosového protokolu se zabalí APDU příkazy a data.

1.3.1 ATR

ATR (*Answer To Reset*) je signál, který karta vyšlé po vložení do čtečky. Jeho obsahem jsou údaje související s předávacím protokolem, informace o kartě a pro navázání a udržení spojení s ní.

1.3.2 APDU

APDU je zkratka z anglického *Application Protocol Data Unit*. Jedná se o mezinárodně standartizovanou komunikační jednotku na aplikační vrstvě. Existují dva formátu APDU zpráv: příkazy a odpovědi.

APDU příkaz se skládá z povinné hlavičky (header) a nepovinného těla (body), jeho struktura je znázorněna na obrázku 1.2.

Hlavička				Tělo		
CLA	INS	P1	P2	Lc	Data	Le

Obrázek 1.2: Struktura APDU příkazu

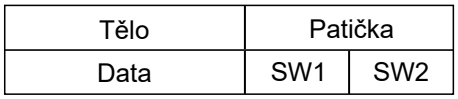
V hlavičce jsou čtyři parametry, každý o velikosti jeden bajt:

- bajt třídy **CLA**
- bajt instrukce **INS**
- bajt prvního parametru **P1**
- bajt druhého parametru **P2**

Bajty třídy a instrukce identifikují operace, která se bude provádět. Tělo APDU příkazu se skládá z:

- pole **Lc** o velikosti jeden bajt, jehož obsahem je počet bajtů v poli dát.
- pole **Data**, které obsahuje Lc bajtů dat.
- pole **Le** o velikosti jeden bajt reprezentuje očekávanou velikost pole Data v odpovědi.

APDU odpověď se skládá z nepovinného těla a povinné patičky, které obsahuje dva bajty Status Words: **SW1** a **SW2**. Jedná se o „návratové kódy“ z čipové karty, které buď signalizují úspěšnost provedeného příkazů, nebo v opačném případě obsahují informace o chybě.



Obrázek 1.3: Struktura APDU odpovědi

Tabulka 1.3 ukazuje základní rozdělení odpovědi podle hodnot SW1 a SW2.

Tabulka 1.3: Návratové kódy SW1 a SW2 podle ISO/IEC 7816-4

		SW1	**SW2**
Operace dokončena	Zpracováno normálně	90 61	00 XX
	Zpracováno s varováním	62 63	XX XX
Operace přerušena	Chyba při zpracování	64 65	XX XX
	Chyba při kontrole	67 – 6F	XX

1.4 GlobalPlatform

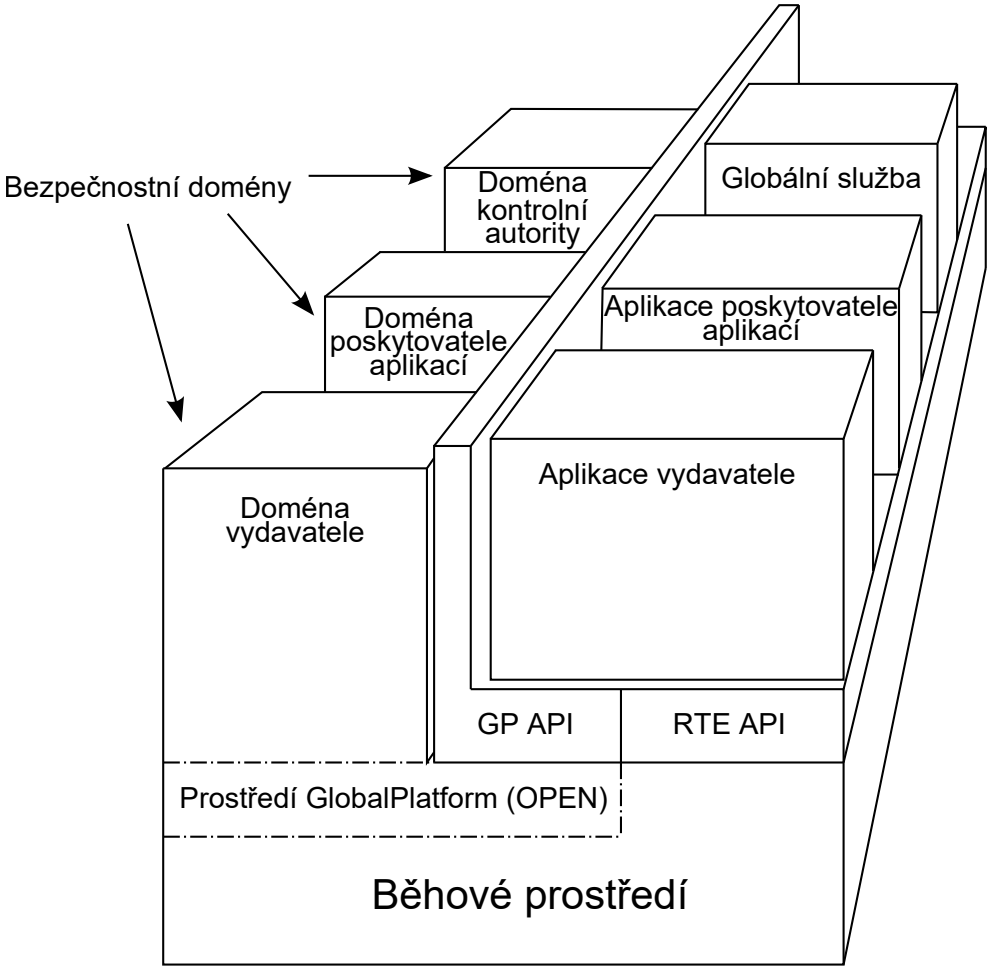
GlobalPlatform je nezávislá nezisková organizace, cílem které je standardizace infrastruktury pro usnadnění vývoje, nasazení a správu čipových karet. Původně byla vytvořena asociací Visa pod názvem Open Platform. Jejich specifikace umožňuje vydavatelům čipových karet vytvářet flexibilní jedno i víceaplikační systémy. Čipové karty, které mají implementován systém podle GlobalPlatform, umožňují používat bezpečný mechanismus nahrávání a instalace aplikací[6] na kartu a zajistit tak řízení životního cyklu multiaplikačních karet. Návrh GlobalPlatform předpokládá, že vydavatel nechce nutně spravovat veškerý obsah karty, zejména ten obsah, který mu nepatří.

Zdrojem informací pro tuto kapitolu je převážně specifikace GlobalPlatform v2.2 [8].

[6]Protoze specifikace GlobalPlatfrom je platformě nezávislá v této kapitole applety jsou ozancovane jako aplikace.

1.4.1 Architektura GlobalPlatform

Architektura GlobalPlatform se skládá z několika komponent, které zajišťují nezávisle na hardwaru a výrobci rozhraní. Obrazek1.4 znázorňuje ukázkovou konfiguraci karty, která obsahuje jednu nebo více aplikací od vydavatele, společníků vydavatele a několik aplikací poskytujících globální služby jiným aplikacím. Všechny aplikací mají být implementované v zabezpečeném běhovém prostředí, které poskytuje hardwarově nezávislé API.



Obrázek 1.4: Architektura čipové karty GlobalPlatform

OPEN je jádro systému GlobalPlatform a jeho nejdůležitější funkce jsou poskytování API aplikacím, výběr aktivní aplikace a směřování příkazů na ni. Také vykonává nahrání aplikačního kódu a s tím spojené správu paměti karty, instalace aplikací nahraných na kartu a má zodpovědnost za dodržení bezpečnostních pravidel definovaných pro řízení obsahu karty.

Bezpečnostní domény jsou privilegované (systémové) aplikace (nikoli místa

v paměti, do kterých se instalují aplikace), které na kartě zastupují vydavatele karty nebo poskytovatele aplikací. Obsahují kryptografické klíče, určené pro vytvoření zabezpečeného kanálu a postřednictvím domén se provádí správa obsahu karty. Aplikace můžou využívat služby bezpečnostních domén pro veškeré procesy spojené s otevíráním a použitím zabezpečeného kanálu. Podle svého určení rozeznáváme 3 typy domén: doména vydavatele, kontrolní autority a aplikační domény. Všechny karty mají povinnou jedinou doménu vydavatele. Poskytovatelé aplikací mohou mít svojí vlastní doménu ke správě svých aplikací, s klíči, které jsou nezávislé na vydavateli karty.

Architektura GlobalPlatform nestanovuje RTE a může být implementována pro jaky-koliv multiaplikacni systém, například Java Card či MULTOS.

1.4.2 Zabezpečení komunikace

Zabezpečení komunikace podle specifikace GlobalPlatform je v souladu s ISO standardy a je jejich nadstavbou. Architektura GlobalPlatform poskytuje zásadní mechanismy, které umožňují zabezpečenou komunikace mezi kartou a off-card entitou, tzv. zabezpečené kanály. Zabezpečeny kanál dovoluje autentikace entit a nastavení klíčů, které budou použité během sessiony. Tedy protokol zabezpečeného kanálu (*Secure Channel Protocol*, dále jen SCP) definuje pravidla navázání a použití zabzepeceneho kanálu. V rámci jakeho-koliv protokolu zabezpečeného kanálu jsou poskytované tří úrovní zabezpečení komunikace mezi off-card entitou a kartou:

- Vzájemná autentikace entit – karta a externí entita ověří totožnost pomocí znalosti sdíleného tajemství.
- Integrita dat – příjemce kontroluje, jestli nedošlo k modifikace pořadí dat a jejich zdrojem je autentikovana entita.
- Důvěrnost dát – data nejsou přístupné neautentikovane entitě.

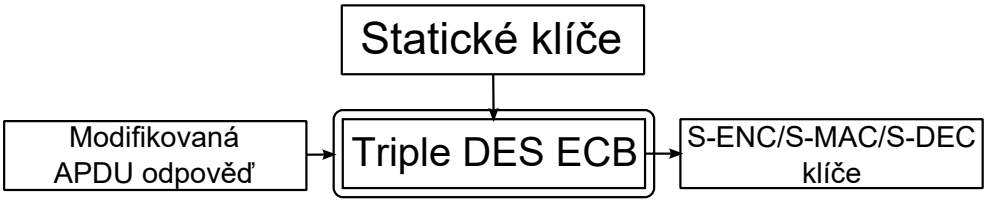
Aplikace můžou přímo ovládat zabezpečeny kanál vlatnictvim sady kryptografických klíčů a plnou implementací protokolu, takové aplikace jsou například bezpečnostní domény. Nebo nepřímo pomocí využití prostředků bezpečnostních domén.

V roce 2006 GlobalPlatform zavedla SCP verze 02 a protokol SCP01 prohlásila jako zavrhovaný.

1.4.2.1 SCP01

SCP01 pro vzájemnou autentikace karty a externí entity používá symetrickou kryptografií. Explicitní inicializace kanálu se začíná APDU příkazem INITIALIZE UPDATE a pomocí odpovědi se vytvářejí tři klíče: šifrovací klíč *S-ENC*, klíč autentikace zpráv *S-MAC*, který je zopdovedny za kontrolu integrity APDU příkazů, a klíč pro šifrování citlivých dat *S-DEC*. Generování

klíčů se provádí prostredstvim TripleDES v ECB módu a je znázorněna na obrázku 1.5. Off-card entita ověří kryptogramu karty a v případě úspěchu odešle APDU příkaz EXTERNAL AUTHENTICATE a karta ověří kryptogramu off-card entity.



Obrázek 1.5: Generování klíčů podle SCP01

Pokud je požadována kontrola integrity APDU příkazů, než příkaz se odešle, původní příkaz se modfikuje a vygeneruje se jeho C-MAC. V CLA bajtů příkazu se nastaví bit identifikující, že APDU příkaz obsahuje zabezpečení, Lc bajt se zvětší o délku C-MAC a na konec se přidá vyegenerovany C-MAC. Stejný postup provede karta a porovnáním C-MAC se zaručuje integrita APDU příkazu. Pro generování C-MAC se používá Triple DES v CBC módu a C-MAC klíč.

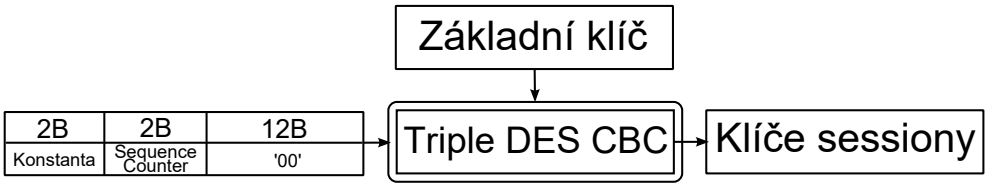
Je-li nastavená úroveň zabezpečení požaduje šifrování pole DATA, pole DATA se šifruje pomocí Triple DES v CBC módu a S-ENC klíčů. Lc bajt se zvětšuje, a na konec se přidá C-MAC.

Požadovaná úroveň zabezpečení se nastavauje v APDU příkazu EXTERNAL AUTHENTICATE v bajtů P1 C.3.

1.4.2.2 SCP02

Stejně jako SCP01 je založen na symetrické kryptografii a explicitní inicializace kanálu se provádí pomocí APDU příkazu INITIALIZE UPDATE a EXTERNAL AUTHENTICATE. SCP02 přináší větší možnosti implementace, například kontrolu integrity odpovědi – R-MAC, což znamená, že sada klíčů sessiony se rozšíří o R-MAC klíč. Struktura pole DATA v odpovědi INITIALIZE UPDATE narozdil od SCP01 obsahuje dva bajty Sequence Counter, které se používají k generování klíčů sessiony. Dále protokol definuje čtyři konstanty pro klíče místo použití statických klíčů:

- *'0101'* – C-MAC
- *'0102'* – R-MAC
- *'0182'* – S-ENC
- *'0181'* – S-DEK

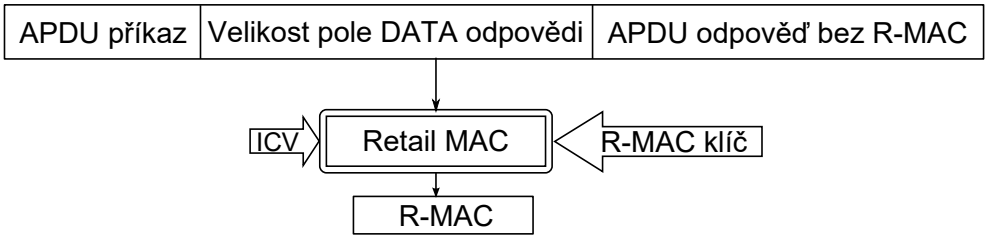


Obrázek 1.6: Generování klíčů podle SCP02

Pro generování klíčů (viz obrázek 1.6) se používá Triple DES v CBC modu, což je další rozdíl oproti SCP01.

Další rozšíření SCP02 je možnost generování C-MAC k nemodifikovanému APDU příkazu. Možnost generace C-MAC k modifikovanému APDU zůstává, viz 1.4.2.1. Pro generování C-MAC se používá algoritmus Retail MAC a C-MAC klíč sessiony. ICV se nastaví na C-MAC, který byl vygenerovaný k předchozímu příkazů a zašifruje se pomocí Single DES. První ICV sessiony použity pro APDU příkaz EXTERNAL AUTHENTICATE je nulový a není šifrovaný.

Pokud nastavená úroveň zabezpečení vyžaduje integritu odpovědi off-card entita musí ukládat poslední APDU příkaz. Příkaz se ukládá nemodifikovaný, ale pokud pole DATA je prázdné do bajtů Lc se zapíše nula.



Obrázek 1.7: Generování R-MAC

Off-card entita generuje R-MAC, který se porovnána s R-MAC z APDU odpovědi, čímž se zaručuje integrita. První ICV je nulový, dále ICV se stává poslední vygenerovaný R-MAC. Na rozdíl od ICV, který se používá pro C-MAC, ICV pro R-MAC se nešifruje.

1.5 NetBeans

Tato kapitola se věnuje platformě NetBeans a NetBeans IDE. Informace použité v této kapitole byly převzaté z knihy Platforma NetBeans: Podrobný průvodce programátora [9].

NetBeans Platform je framework, který umožňuje vytváření velkých desktop aplikací. Jeho rodičem je projekt českých studentů Xelfi – vývojové prostředí pro jazyk Java, které pak bylo přejmenované na NetBeans. Časem se projekt NetBeans rozrůstal a vydělením obecné části, společné pro desktopové

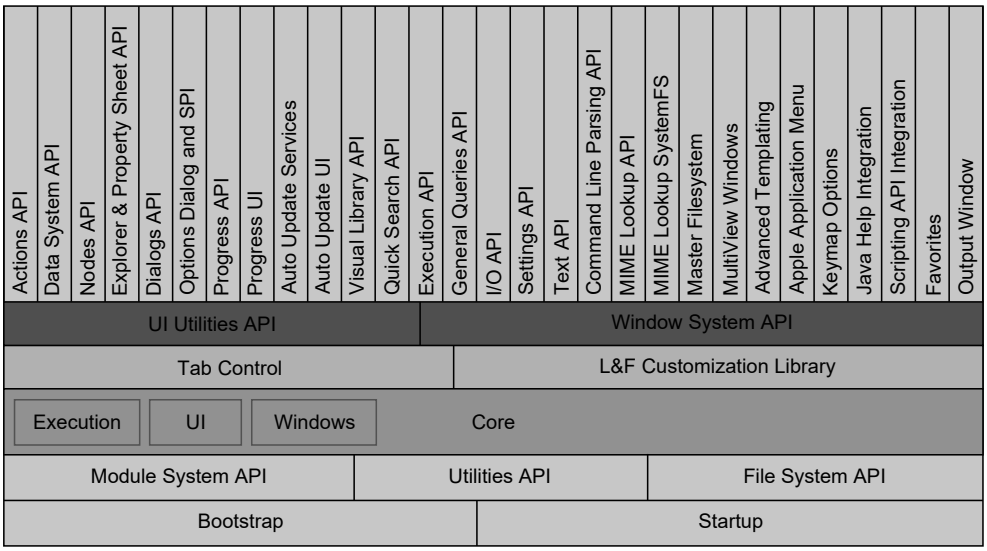
aplikace, vznikla platforma NetBeans. Stejnojmenné Nebeans IDE nyní je postaveno na platformě Netbeans.

1.5.1 Netbeans Platform

Platforma je postavena na standardní knihovně Java Swing a pokrývá především následující oblasti vývoje:

- Správa oken a vizualizace dat
- Správa akcí a jejich provázání s uživatelskými nabídkami
- Správa modulů a jejich závislostí
- Internacionalizace
- Perzistence dát

NetBeans Platform má modulární architekturu, kterou demonstruje obrázek 1.8.



Obrázek 1.8: Architektura NetBeans Platfrom

Jádrem celé platformy je běhový kontejner NetBeans (*NetBeans Runtime Container*), který je tvořen pěticí modulů:

- Bootstrap – nejdříve spouštěný modul. Zpracovává parametry příkazové řádky, stará se o nalezení běhového prostředí Java a inicializaci počátečního zavaděče tříd (boot classloader), který načte modul Startup.

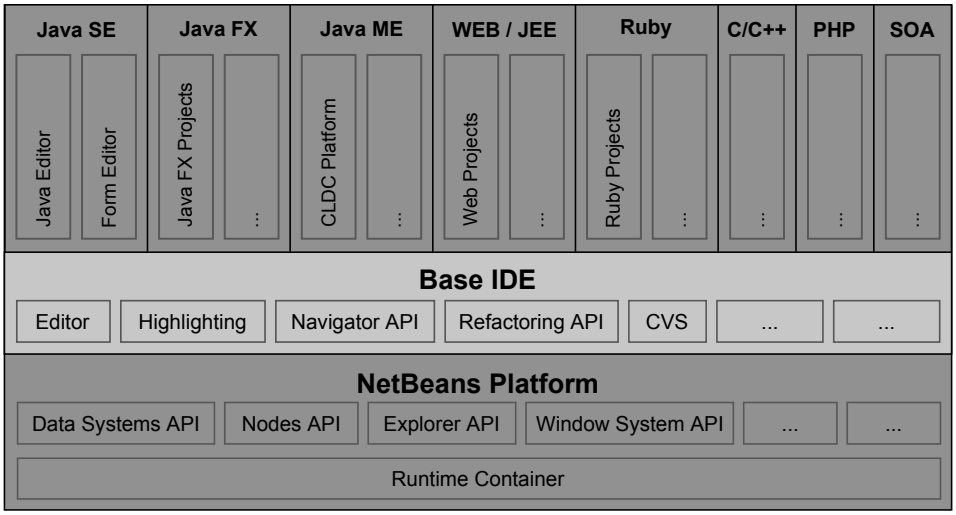
- Startup – má za úkol nastartovat aplikací. Inicializuje správce modulu (Module System) a souborového systému (File System).
- Module System – spravuje moduly a jejich závislosti. Informuje běhový kontejner, jak má jednotlivé moduly načítat.
- File System – realizuje virtuální souborový systém, kterým platforma sjednocuje přístup k datům.
- Utilities – obsahuje pomocné komponenty například pro práci s XML soubory.

Úkolem běhového kontejneru je pouze připravit prostředí pro ostatní moduly. Proto, pokud při spouštění nenalezne žádné moduly, po inicializaci skončí. Nalezeným modulům běhový kontejner vyřeší závislosti a pokud jsou splněny, je modul načten. Moduly nemusí být načteny ihned při startu, ale až když jsou potřeba.

Jak je vidět na obrázku 1.8 sada dalších API a služeb je také obsahem distribuce NetBeans Platform.

1.5.2 Netbeans IDE

Netbeans IDE je multiplatformni open-source vývojové prostředí vlastněné firmou Oracle. Jak už bylo řečeno její technologickým základem je platforma Netbeans. Tedy také má modulární architekturu **??** a díky tomu je snadno rozšířitelne a může se přizpůsobit podle potřeb jednotlivých uživatelů.



Obrázek 1.9: Architektura NetBeans IDE

Základní funkcionalita obsahuje grafický debugger, obsluhu automatizačních nástrojů, statickou analizu. Syntaktická kontrola kódu a nápovědy jsou samozřejmostí. V současné době aktulani verze Netbeans IDE je 8.2, která byla představena v roce 2016.

1.6 Alternativní vývojové nástroje

Existuje řada nástrojů pro usnadnění práce s Java Card applety. Jedné umožňují proces vývojů, napr: Java Card Development Kit, EclipseJCDE nebo JCKit. Jiné, jako GPShell, jsou zaměřené především na komunikaci s kartou nebo správu její obshau. V této kapitole jsou popsané volně dostupné nástroje pro vývoj appletu na platformě Java Card.

1.6.1 Java Card Development Kit

Java Card Development Kit (JCDK) poskytuje kompletně a nezávislé vývojové prostředí, ve kterém Java Card applety můžou být vyvinuté, otestované a nasazeny. Součástí JCDK jsou:

- Sada CLI nástrojů pro vývoj a nasazení aplikací. Jsou nezbytnou části JCDK jejich podrobný popis lze nalézt v sekci1.6.1.1.
- Komponenty pro emulace (včetně kryptografické funkcionality). Vyhovují specifikace Java Card Platform.
- Ukázkové zdrojové kódy a dokumentace.

1.6.1.1 Nástroje Java Card Development Kitu

Táto sekce popisuje jednotlivé nástroje z JCDK a jejih funkcionalitu.[7]

Jeden z největších přínosů verze 3.0.5. JCDK je Eclipse plug-in, který výrazně zjednodušuje použití ostatních. Doporučená pro tento plug-in verze prostředí Eclipse Luna již není v dnešní době aktuální, což může být problematické pro některé vývojáře.

- **Eclipse plug-in** – Spuštění zbytku nástrojů v Eclipse IDE[8].
- **apdutool** – Odesílání APDU příkazy do běhového prostředí Java Card nebo simulátoru.
- **capdump** – Vytváření ASCII reprezentace `.cap` souborů.

[7] Seznam nástrojů a jejich funkcionalita je aktuální pro klasickou edice Java Card platformy verze 3.0.5.

[8] Vývojové prostředí určené pro programování v jazyce Java. Je rozšířitelne za pomocí pluginů.

- **capgen** – Generování `.cap` souborů z Java Card Assembly souborů.
- **classic_simulator** – Debugování Java tříd v Eclipse IDE nebo z příkazové řádky.
- **Converter** – Konvertace `.class` souborů do `.cap`.
- **cref** – Spuštění C-language Java Card RE referenční implementace z příkazové řádky. Existují tři verze tohoto nástrojů pro ovládání různých komunikačních protokolů.
- **exp2text** – Vytváření textové reprezentace `.exp` souborů.
- **on-card installer** – Instalace applikace a stažení Java Card balíčků. Také je schopný smazávat balíčky a applety z karty.
- **maskgen** – Maskování Java Card Assembly souborů.
- **Normalizer** – Zpětná kompatibilita s applety v.2.2.2.
- **off-card verifier** – Ověření výstupních souborů podle specifikací Java Card. Skládá se ze třech nástrojů: verifycap verifikuje `.cap` soubory, verifyexp – `.exp` a verifyrev kontroluje binární kompatibilitu porovnáním příslušných `.exp` souborů.
- **scriptgen** – Převod `.cap` souborů do sekvence APDU příkazů.

1.6.2 EclipseJCDK

EclipseJCDK je plugin pro Eclipse IDE a používá plafromu Eclipse pro obalení JCDK. Má dvě zásadně nevýhody: podporuje JCDK pouze ve verze 2.2.2 a poslední aktualizace tohoto pluginu se provedla v roce 2012[10]. Vzhledem k tomu, že JCDK má vlastní nástroj pro integraci do Eclipse IDE, plugin EclipseJCDK je zastaralý a již není nadále použitelný.

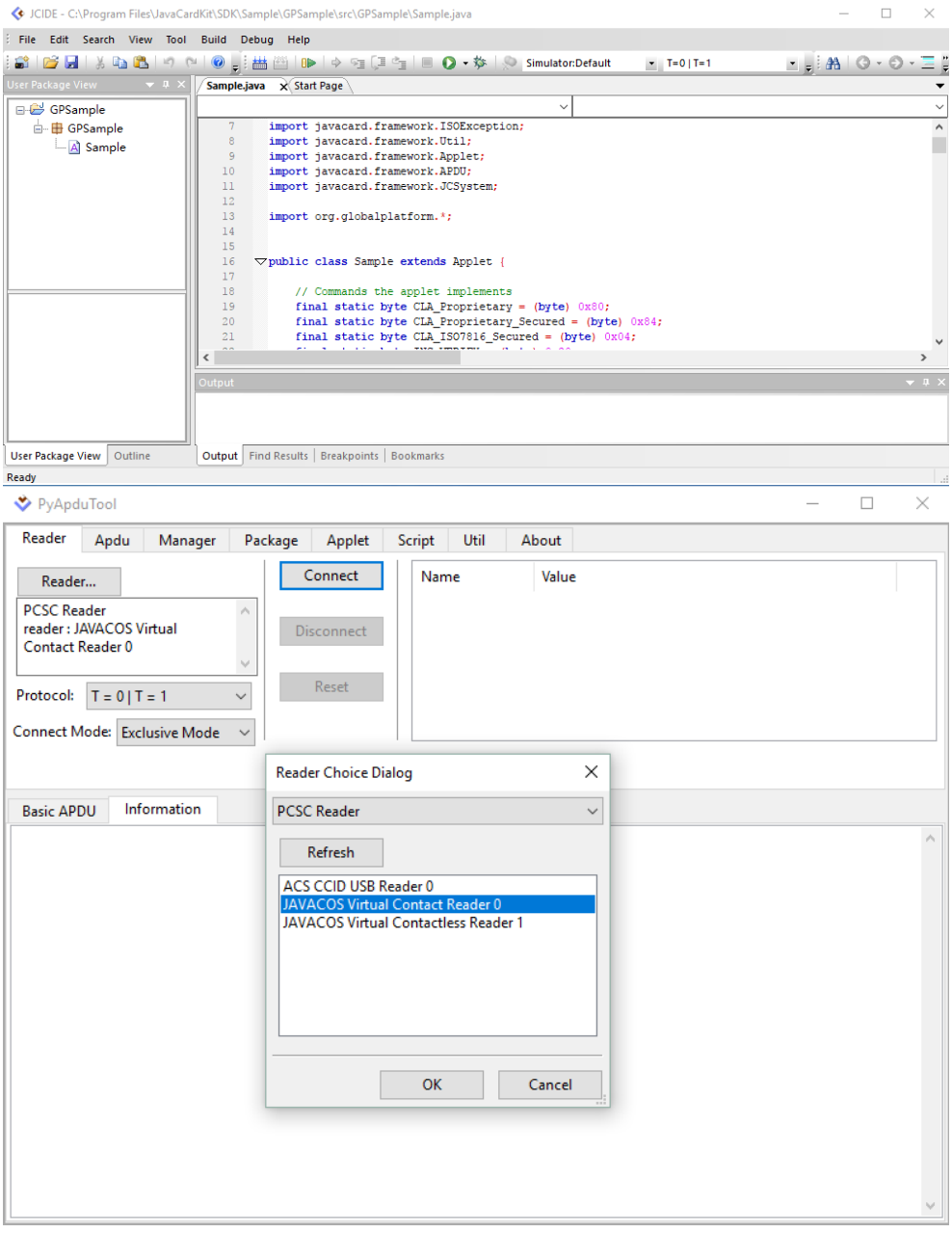
1.6.3 jCardSim

Je multiplatformní open-source simulator[11], implementuje Java Card API v.2.2.1/2. Zrychluje vývoj prototypů appletu, APDU skriptů, verifikačních a unit testů. Simulátor API poskytuje možnost práce jako s reální Java kartou s použitím `javax.smartcardio.*`. Na rozdíl od Java Card Development Kit dovoluje spuštění aplikace bez nutnosti konvertace do `.cap` souborů a má odlišnou realizaci `javacard.security.*`. APDU skripty vytvořené pomocí jCardSim jsou kompatibilní s apdutool z Java Card Development Kit.

1.6.4 JCKit

JCKit se skládá z JCIDE a PyApduTool[12]. JCIDE je vývojové prostředí, které bylo navrženo speciálně pro vývoj v jazyce Java Card. Poskytuje standardní automatické doplňování kódu a umožňuje debugovani v grafickém prostredi. pyApduTool je nástroj, který umožňuje komunikaci s kartou pomocí čtecího zařízení připojeného k počítači, správu balíčků a appletu na kartě a provedení integračních testu. Stejně jako Java Card Development Kit JCKit obsahuje vše potřebné pro vývoj a nasazení appletu, ale na rozdíl od JCDK je kompatibilní pouze s Windows OS. Jedním z největších důvodu použití JCKit je jeho pravidelná aktualizace[9]. Další výhodou JCKit je přehledný user-friendly UI a snadná instalace.

[9] Vydání poslední verze proběhl v roce 2016.



Obrázek 1.10: JCKit

Kapitola 2

Návrh

Kapitola popisuje požadavky, které jsou kladené na moduly a návrh změn. Jelikož spousta věcí, které se týkaly struktury sady a vazby modulů, již byla vyřešena v Ing. Petrem Vlaskem[1] a Bc. Filipem Munzarem[2] tato část je stručná.

2.1 Model požadavků

2.1.1 Funkční požadavky

1. Opakované nahrávání `.cap` souborů.

2. Opakovaná instalace appletu.

3. Správa obsahu Java karet podporujících SCP02.

2.1.2 Nefunkční požadavky

1. Podpora karty NXP J3A.

2. Podpora současných verzí vývojového prostředí NetBeans IDE – sada modulů musí fungovat na nejnovějších verzích NetBeans IDE.

2.2 Návrh změn

Od roku 2014 vývoj NetBeans IDE pokročil a proto aktualizace modulů je jedním z cílem této práci. Jako prvním krokem bylo zvoleno provést testování původní sady v Netbeans IDE verze 8.2, což by mělo zkontrolovat je-li sada funkční a dát přehled budoucích změn.

2.2.1 Testování

Původní sada pluginů byla [2] nainstalována do NetBeans IDE 8.2 a byly provedeny uživatelské testy.

2.2.1.1 Java Card Project

Postupně byly vytvořeny nové projekty a projekty z existujících zdrojů, které využívali JCDK 2.1.2, 2.2.2, 3.0.4 a 3.0.5. Založení všech projektů proběhlo bez problémů. Dále projekty byly úspěšně převedeny do `.cap` souborů pomocí akcí **Build and Convert Project**. Zdrojové soubory použité pro vytvoření prokektu z existujících zdrojů jsou umnistene na přiloženém CD ve složce `tests/sources`.

2.2.1.2 Java Card Manager

K testování tohoto modulu byly použity karty GemXpresso R4 72K, GemCombiXpresso R4 72K, GemXpresso Pro R3.2 64K a NXP J2A 080. Bylo otestováno připojení, odpojení a nahrání appletu a load souborů. Nahrání selhalo pouze ve chvíli, kdy byl nahráván soubor appletu s verzí JCDK, kterou karta nepodporovala, což v podstatě je správné chování. Jinak proběhlo vše v pořádku K testování byly použity `.cap` soubory vytvořené v rámci práci Bc. Filipa Munzara[2] a v minulém kroku.

2.2.1.3 Java Card APDU Manager

Bylo otestováno spuštění APDU Manageru a odesílání APDU příkazu. Pokud příkaz měl správnou syntaxi, v odpovědi karty položka SW se rovnala `0x9000`, což znamená úspěch. Na špatné příkazy karta vracela chybové kódy. Také bylo otestováno chování modulu v případě odpojení resp. vyjmutí karty. APDU Manager správně ukázal odpojení karty.

2.2.1.4 SimpleSmartCard API

Spuštění unit testů `testAppletConverter` se skončilo úspěchem.

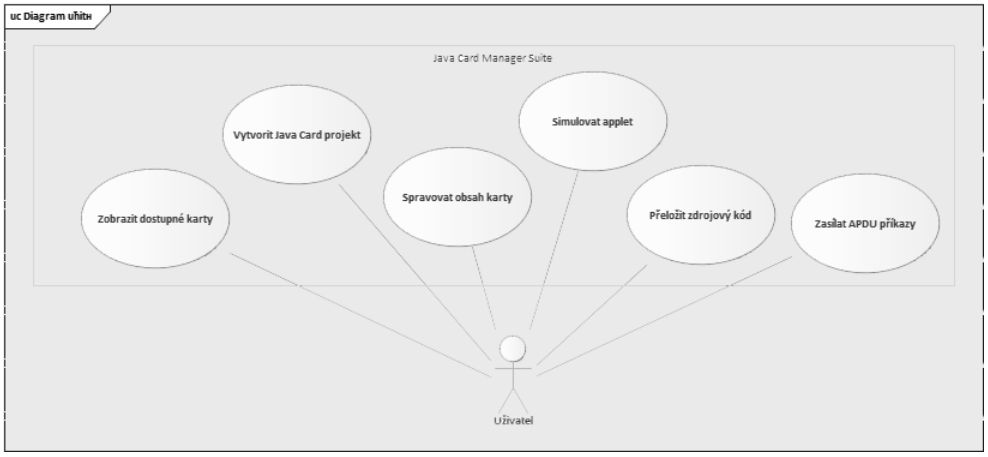
2.2.2 Plánované změny

Předchozí krok úplně zopakoval testování, které bylo provedeno Bc. Filipem Munzarem[2], což ověřilo, že moduly jsou plně funkční v aktuální verzi Netbeans IDE. Tudíž žádné změny, které by zachovali původní funkčnost, nejsou nutné.

Aby Java Card Manager byl schopen připojit chytrou kartu s SCP02 do modulu SimpleSmartCard API bude přidána implementace abstraktní třídy

`CardManagerAdapter`, která bude zodpovědná za správu karty podle specifikace GlobalPlatform v2.2. Tato změna také by měla ovlivnit UI modulu Java Card Manager.

Panely pro nastavení instalace appletu a vytvoření LOAD souborů budou upravené, aby opakovaně nahrávání bylo zjednodušeno.



Obrázek 2.1: Diagram užití

Kapitola 3

Implementace

Tato kapitola popisuje implemetaci jednotlivých požadavků.

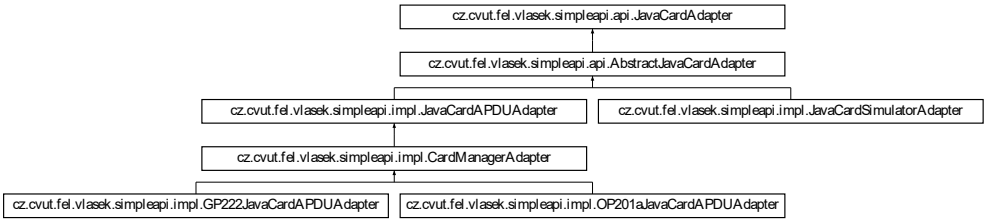
3.1 Podpora SCP02

Sekce popisuje změny se týkající přidání podpory protokolu SCP02 a související s tím změny v UI.

3.1.1 GP222JavaCardAPDUAdapter

Do balíčku `cz.cvut.fel.vlasek.simpleapi.impl` byla přidána třída `GP222JavaCardAPDUAdapter`, která implentuje rozhraní `CardManagerAdapter` podle i=15[10] SCP02 a kontrolu integrity odpovědi navíc:

- Explicitní inicializace zabezpečeného kanálu
- Tři bezpečnostních klíče
- Generování C-MAC na modifikovaný APDU příkaz
- Šifrování ICV
- Podpora R-MAC



Obrázek 3.1: Hierarchie tříd

[10]Sada parametrů, které musí implementovat bezpečnostní doména karty.

Metoda `authenticate(int securityLevel)` má na starosti explicitní inicializace zabezbeceneho kanálu a generování bezpečnostních klíčů. Jako argument dostává požadovanou úroveň zabezpečení. Vzhledem k tomu, že SCP02 přináší R-MAC, enum `SecurityLevels` byl rozšířen a již existující položky byly upravené, aby jejich název a popis odpovídal specifikaci GlobalPlatform v2.2.

Zdrojový kód 3.1: Úrovní zabezpečení

```
public static enum SecurityLevels {
    NO_SECURITY(0x00, "NO_SECURITY"),
    C_MAC(0x01, "C_MAC"),
    C_DECRYPTION_C_MAC(0x03, "C_DECRYPTION + C_MAC"),
    R_MAC(0x10, "R_MAC"),
    C_MAC_R_MAC(0x11, "C_MAC + R_MAC"),
    C_DECRYPTION_C_MAC_R_MAC(0x13, "C_DECRYPTION + C_MAC + R_MAC"
        );
    ...
}
```

Jelikož R-MAC není součástí i=15, karta nemusí ho podporovat, a v APDU odpověď bude obsahovat informace, že bajt P1 APDU příkazu EXTERNAL AUTHENTICATE není validní. V tomto případě metoda `authenticate` vyhodí výjimku a uživatel bude informován, že požadovaná úroveň zabezpečení není poskytovaná kartou.

Zdrojový kód 3.2: Kontrola podpory R-MAC kartou

```
if (response.getSW() == 0x6A86) { // Status Word 6A86 oznacuje
    nevalidni bajt P1 APDU prikazu
    throw new AdapterException("External Autheticate: Security
        Level is unsupported by card");
}
```

Pokud se podařila inicializace modifikace APDU příkazu v souvisloti s nastevenou úrovni provádí privátní metoda `wrapCommand(CommandAPDU command)`.

Pokud byl zvolen R-MAC, metoda `unwrapResponse(ResponseAPDU response)` je zodpovědná za integritu odpovědi. V případě chyby vyhazuje výjimku a informuje uživatele, že intergrita odpovědi byla porušena.

Protože pro generování C-MAC nebo R-MAC SCP02 používá definovány v ISO/IEC 18033-3 Single DES Plus Final Triple DES neboli tzv. "Retail MAC"do třídy `CryptoServices` byla přidána jeho implentaci.

Zdrojový kód 3.3: Retail MAC

```
public static void singleDESTripleDES(byte[] key, byte[] data,
    byte[] vector, byte[] encData){
   ... //inicializace klicu a vektoru
   Cipher cbc1 = Cipher.getInstance("DES/CBC/NoPadding");
   Cipher cbc2 = Cipher.getInstance("TripleDES/CBC/NoPadding");
```

```
    cbc1.init(Cipher.ENCRYPT_MODE, secretKeyDes, new
        IvParameterSpec(vector));
    cbc2.init(Cipher.ENCRYPT_MODE, secretKeyTripleDes, new
        IvParameterSpec(vector));

    byte[] intermeidateResult;

    if(data.length > 8){
         intermeidateResult = cbc1.doFinal(data, 0, data.length -
             8);
         System.arraycopy(intermeidateResult, intermeidateResult.
             length-8, encData, 0, 8);
         cbc2.init(Cipher.ENCRYPT_MODE, secretKeyTripleDes, new
             IvParameterSpec(encData));
     }

    intermeidateResult = cbc2.doFinal(data, data.length - 8, 8);
    System.arraycopy(intermeidateResult, intermeidateResult.length
        - 8, encData, 0, 8);
}
```

Dalším krokem by měla být implementace abstraktních metod definovaných rozhraním `CardManagerAdpater` zodpovědných za správu obsahu karty. Jelikož třída `OP201aJavaCardAPDUAdapter` už obsahovala jejich realizace, která byla v souladu se specifikace GlobalPlatform v2.2, implementace následujících metod byla přesunuta do společného rodiče `CardManagerAdpater`:

- `delete`
- `getData`
- `getStatus`
- `installForInstall`
- `installForLoad`
- `load`
- `loadArrayOfBytes`

Aby se zlepšila logická struktura kódu do balíčku `cz.cvut.fel.vlasek.simpleapi.impl` byla přidána třída `JavaCardAdapterEnums`. Její obsahem jsou enum třídy, které původně patřily do `OP201aJavaCardAPDUAdapter` a jsou společné pro potomky `JavaCardAdapter`.

3.1.2 Připojování karty a UI

Protože verze SCP karty se zjišťuje až v odpovědi na příkaz INITIALIZE UPDATE, panel `ConnectCardPanel` byl rozšířen o volbu protokolů. Také do `ConnectCardPanel` byly přidané UI elementy, které se týkají SCDP02. Pokud

uživatel vybere SCP01, aby se nezobrazovaly úrovní zabezpečení, které tento protokol nepodporuje `jComboBox_protocol` byl nastaven `ActionListener`:

Zdrojový kód 3.4: Dynamická změna obsahu ConnectCardPannel

```
jComboBox_protocol.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e) {
        javax.swing.JComboBox<String> source = (javax.swing.
            JComboBox<String>) e.getSource();
        String selectedValue = (String)source.getSelectedItem();
        if(selectedValue.equals("SCP01")){
            jComboBox_securityLevel.setModel(secSCP1);
            ...
            jTextField_kek.setEnabled(true);
        } else if(selectedValue.equals("SCP02")) {
            jComboBox_securityLevel.setModel(secSCP2);
            ...
            jTextField_kek.setEnabled(false);
        }
    }
});
```

Kde `secSCP1` a `secSCP2` jsou členy `ConnectCardPannel` typu `DefaultComboBoxModel<JavaCardAdapterEnums.SecurityLevels>`. Jejich obsahem jsou úrovní zabezpečení které podporuje SCP01 resp. SCP02.

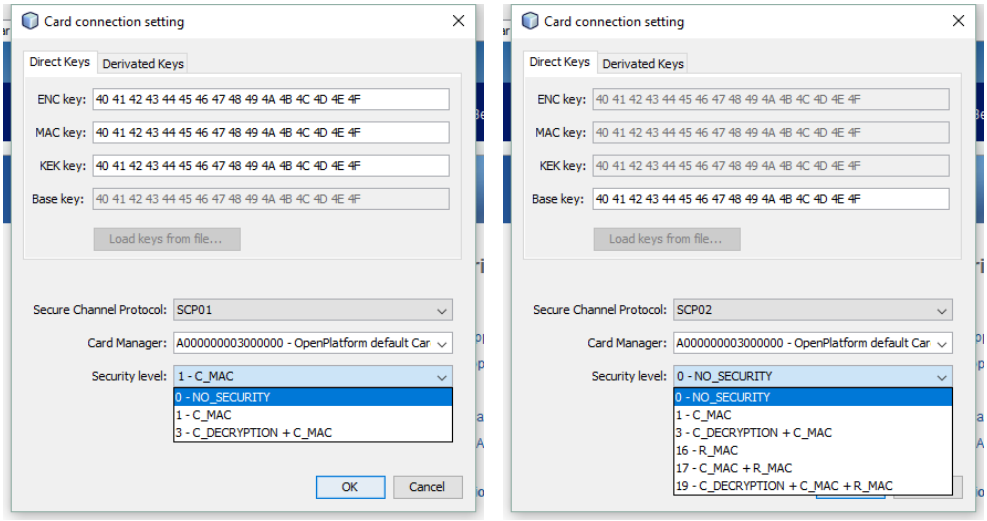
Zdrojový kód 3.5: Inicializace úrovní zabezpečení

```
public ConnectCardPanel() {
    // SCP01 supported security levels
    secSCP1 = new DefaultComboBoxModel<JavaCardAdapterEnums.
        SecurityLevels>();
    secSCP1.addElement(JavaCardAdapterEnums.SecurityLevels.
        NO_SECURITY);
    secSCP1.addElement(JavaCardAdapterEnums.SecurityLevels.C_MAC)
        ;
    secSCP1.addElement(JavaCardAdapterEnums.SecurityLevels.
        C_DECRYPTION_C_MAC);

    // SCP02 supported security levels
    secSCP2 = new DefaultComboBoxModel<JavaCardAdapterEnums.
        SecurityLevels>(JavaCardAdapterEnums.SecurityLevels.
        values());
    ...
}
```

`ActionListener` také má na starosti blokaci elementů UI, které nejsou zapotřebí pro zvolený protokol.

V závislosti na zvoleném protokolu ve třídě `JavaCardAPDUConnections` se vytvoří instance buď `OP201aJavaCardAPDUAdapter` nebo `GP222JavaCard-APDUAdapter`.



Obrázek 3.2: Připojení karty

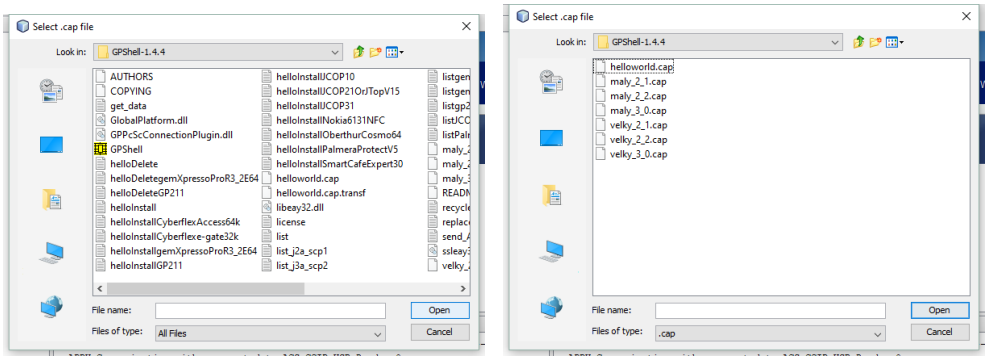
3.2 Opakované nahrání .cap souborů

První oprava se netýká opakovaného nahrání a byla záměrná na usnadnění vyhledávání `.cap` souborů.

Zdrojový kód 3.6: Filtrace souborů

```
public static void selectFileDialog(JTextField textField, JPanel
    owner) {
    JFileChooser chooser = new JFileChooser();
    FileNameExtensionFilter filter = new FileNameExtensionFilter(
        ".cap", "cap");
    chooser.setDialogTitle(NbBundle.getMessage(
        JavaCardManagerHelper.class, "JavaCardManager.BrowseTitle
        "));
    chooser.setFileFilter(filter);
    String path = textField.getText();
    if (path.length() > 0) {
        File f = new File(path);
        if (f.exists()) {
            chooser.setSelectedFile(f);
        }
    }
    if (JFileChooser.APPROVE_OPTION == chooser.showOpenDialog(
        owner)) {
        File projectDir = chooser.getSelectedFile();
        textField.setText(projectDir.getAbsolutePath());
    }
}
```

Díky nasazení filtru průzkumník zobrazuje pouze `.cap` soubory. Možnost zobrazení všech souborů zůstává.



Obrázek 3.3: Výběr souboru

Další krok byl zaměřen přímo na usnadnění opakovaného nahrání `.cap` souborů.

Nahrání souboru se koná pomocí buď akce `JavaCardManagerLoadAppletAction` nebo `JavaCardManagerInstallAppletAction`, které vyžadují nastavení ze strany uživatele. Pro jejich vyplnění slouží dialogové okna, tvořené `InstallAppletPanel` a `LoadAppletPanel`. Protože akce vždycky vytváří novou instanci panelu, uživatel musí znovu zadat veškeré údaje. Řešením je uložení naposled použitích nastavení a jejich zobrazení v UI, aby uživatel mohl zkontrolovat nebo opravit parametry. Do každé akce byly přidány privátní členy, do kterých se ukládají již použité nastavení. Předání nastavení do příslušného panelů zjednodušuje opakovaně nahrání.

Zdrojový kód 3.7: Předání nastavení do InstallAppletPanel

```
public void setLastParameters(AppletInstallParams
    lastInstallParameters, String lastFile, int lastManager){
    jTextField_file.setText(lastFile);
    jTextField_installParam.setText(lastInstallParameters.
        getInstallationParams());
    ...
    jComboBox_manager.setSelectedIndex(lastManager);
    dialogDescriptor.setValid(valid()); // validace, jestli
        naposled pouzite parametry jsou stale aktualni
}
```

Kapitola 4

Testování

Testování modulu mělo dvě cíle:

1. Ověření, jestli nebyla poškozena původní funkcionalita
2. Otestování práce modulu s kartou typu NXP J3A.

Testování proběhlo v NetBeans IDE ve verzi 8.2. Pro splnění prvního bodu byly zopakované uživatelské testy popsané v kapitole 2.2.1. Výsledkem všech testů je úspěch. Také bylo otestované připojení karet podporujících SCP01 s použitím SCP02 a naopak. Java Card Manager Suite informuje uživatele, že zvolený protokol není podporován kartou. V průběhu testování modulu Java Card Manager a Java Card Project bylo ověřeno, že proces opakovaného nahrání `.cap` souborů je jednodušší.

4.1 NXP J3A

Jedná se o kartu s dvojitým rozhraním, která používá Java Card verze 2.2.2 a v závislosti na konfiguraci obsahuje buď 40KB nebo 80KB EEPROM paměti[13]. K testování byla využita karta NXP J3A 080.

4.1.1 Java Card Manager

Kartu se dá úspěšně připojit. Avšak pokud nastavená úroveň zabezpečení bude vyžadovat kontrolu integrity odpovědi připojení selže. Je to způsobeno tím, že karta je kompatibilní se specifikací GlobalPlatform v2.1.1, když R-MAC byl zaveden ve verzi 2.2.

Dále byla otestována správa obsahu karty. Na kartu se dá úspěšně nahrát LOAD soubory různé velikosti a vytvoření instance appletu z nahraných LOAD souboru také proběhlo v pořádku. Instalace appletu pomocí akce **Install .cap file** také proběhla bez problémů. Akce **Delete applet** a **Delete file** odstraňují z karty applety a LOAD soubory bez bez jakýchkoliv problémů.

4.1.2 Java Card APDU Manager

APDU manager se dá spustit pro připojenou kartu a modul správně reaguje na odpojení nebo vyjmutí karty. Také bylo otestované odesílání APDU příkazů a příjem odpovědi. V případě správného příkazu Status Word v odpovědi se rovnal `0x9000` jináč nějakému chybovému kódu, viz 1.3.

Zdrojový kód 4.1: Testování APDU komunikace

```
COMMAND: 80 CA 9F 7F [GET DATA]
RESPONSE: 9F 7F 2A 47 90 51 68 47 91 00 78 ...
STATUS: 90 00 [SW OK]

COMMAND: 00 A4 04 00 08 D0 D1 D2 D3 D4 D5 01 01 [SELECT]
RESPONSE: 48 65 6C 6C 6F 20 57 6F 72 6C 64 21
STATUS: 90 00 [SW OK]

COMMAND: 80 CA 00 00 [GET DATA]
RESPONSE: empty response
STATUS: 6A 88 [SW ERROR_DATA_NOT_FOUND]

COMMAND: FF FF FF FF
RESPONSE: empty response
STATUS: 6E 00 [SW ERROR_WRONG_CLA]
```

Dále byla otestována komunikace s appletem na kartě. Na kartu byl nahrán `.cap` soubor vytvořený pomocí Java Card Project ze zdrojových kódů appletu, který ukládá a vrací data. Zdrojové kódy pro tento test byly převzaté z práci Bc. Filip Munzara[2]. I zde vše proběhlo bez problémů.

Zdrojový kód 4.2: Testování APDU komunikace

```
COMMAND: 00 A4 04 00 06 A0 B0 C0 D0 F0 E1 [SELECT] // vyber
    appletu
RESPONSE: empty response
STATUS: 90 00 [SW OK]

COMMAND: 80 01 00 00 03 12 34 56 // nahrani dat
RESPONSE: empty response
STATUS: 90 00 [SW OK]

COMMAND: 80 00 00 00 03 // nacteni dat
RESPONSE: 12 34 56
STATUS: 90 00 [SW OK]
```

Závěr

Mým zadáním bylo doplnění modulu, aby bylo možno vyvíjet applety pro Java karty, které podporují SCP02. Dalšími úkoly byly aktualizace sady na poslední verzi NetBeans IDE a zdokumentovat volně dostupné nástroje pro vývoj appletu na platformě Java Card.

V první fázi bylo provedeno testování sady v NetBeans IDE ve verzi 8.2, během kterého bylo zjištěno že Java Card Manager Suite je plně funkční v aktualniu verzi vývojového prostředí a tudíž žádné změny cílené na aktualizace nejsou nutné.

Druhá fáze se věnovala zkoumání standardu pro chytré karty a komunikace s nimi, především se specifikacemi GlobalPlatform, resp. Open Platfrom. Také byl vytvořen přehled již existujich nástrojů.

Třetí byla cílena na implementaci podpory SCP02 v souladu se specifikaci Globál Platform v2.2. a drobná zlepšení UI. Také jsem se pokusila provést refaktoring aby se zlepšila logická struktura a čitelnost kódu, coz by melo usnadit další údržbu sady.

Sadu Java Card Manager Suite by bylo dobré nadále aktualizazovat. Během zkoumání již existujících nástrojů pro práce s Java kartami jsem nenalezla žádné jiné, které by byly integrované do aktuialni verze NetBeans IDE, a proto by bylo dobrý zveřejnit plug-iny na NetBeans Plugin Portal.

Během této práci jsem se seznámila s několika technologií, ale nejvíc mě zaujaly technologie Java Card, MULTOS a standardisace systému chytrých karet.

Literatura

[1] Vlášek, P.: *Demonstrace šifrování na platformě Java Card [online]*. Diplomová práce, Diplomová práce, České vysoké učení technické v Praze, 2008, [cit. 2015-12-17]. Dostupné z: `https://dip.felk.cvut.cz/browse/pdfcache/vlasep1_2008dipl.pdf`

[2] Munzar, F.: *Podpora vývoje aplikací pro cipové karty Java Card [online]*. Diplomová práce, Bakalárská práce, České vysoké učení technické v Praze, 2014, [cit. 2015-12-17]. Dostupné z: `https://dip.felk.cvut.cz/browse/pdfcache/munzafil_2014bach.pdf`

[3] Chen, Z.: *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, první vydání, 2000, ISBN 0201703297.

[4] Oracle Corporation: Java Card 3 Platform Virtual Machine Specification, Classic Edition Version 3.0.5 [online]. [cit. 2017-04-09]. Dostupné z: `https://docs.oracle.com/javacard/3.0.5/JCVMS/JCVMS.pdf`

[5] Oracle Corporation: Java Card Platform Runtime Environment Specification, v3.0.5 [online]. [cit. 2017-06-18]. Dostupné z: `http://www.oracle.com/technetwork/java/embedded/javacard/downloads/index.html`

[6] MULTOS Limited: Secure multi-app OS for smart devices [online]. [cit. 2017-04-15]. Dostupné z: `http://www.multos.com/`

[7] Spáčil, V.: *Užití čipové karty ke generování a správě šifrovacích klíčů [online]*. Diplomová práce, Bakalárská práce, Masarykova univerzita, fakulta informatiky, 2009, [cit. 2017-12-17]. Dostupné z: `https://is.muni.cz/th/208177/fi_b/?id=162832`

[8] GlobalPlatform, Inc.: Global Platfrom Card Specification Version 2.2 [online]. [cit. 2017-06-27]. Dostupné z: `https://www.globalplatform.org/`

[9] Böck, H.: *Platforma NetBeans: Podrobný průvodce programátora*. Brno: Computer Press, první vydání, 2010, ISBN 978-80-251-3116-9.

[10] Youssef, A.: EclipseJCDE [online]. [cit. 2017-05-08]. Dostupné z: `http://eclipse-jcde.sourceforge.net/`

[11] LICEL LLC: JCardSim: Java Card Runtime Environment Simulator [online]. [cit. 2017-05-08]. Dostupné z: `http://jcardsim.org/`

[12] JAVACOS Technologies Co. Ltd: Java Card Development Kit [online]. [cit. 2017-05-08]. Dostupné z: `http://www.javacos.com/developmentkit.php`

[13] NXP® Semiconductors N.V.: Product Selector Guide[online]. [cit. 2017-06-27]. Dostupné z: `http://www.nxp.com/docs/en/product-selector-guide/939775017001_v9_HR.pdf?fsrch=1&sr=7&pageNum=1`

Příloha **A**

Seznam použitých zkratek

AID Application Identifier

APDU Application Protocol Data Unit

API Application Programming Interface

ATR Answer To Reset

CBC Cipher Block Chaining

COS Card Operating System

DES Data Encryption Standard

ECB Electronic Codebook

EPROM Erasable programmable read-only memory

EEPROM Electrically erasable programmable read-only memory

ICC Integrated Circuit Card

ICV Initial Chaining Vector

ISO International Organization for Standardization

MAC Message Authentication Code

PROM Programmable Read-Only Memory

RAM Random Access Memory

RFID Radio Frequency Identification

ROM Read-only Memory

RTE Runtime Environment

SCP Secure channel protocol

UI User Interface

Příloha **B**

Obsah přiloženého CD

- `documentation`
  - `Java Card Project` ..... dokumentace modulu Java Card Project
  - `Java Card Manager` ..... dokumentace modulu Java Card Manager
  - `Java Card APDU Manager` .... dokumentace modulu Java Card APDU Manager
  - `SimpleSmartCardAPI` .... dokumentace modulu SimpleSmartCardAPI
- `JavaCardManagerSuite` ..... složka s moduly pro NetBeans IDE v8.2
- `src`
  - `impl` ..... zdrojové kódy sady JavaCardManagerSuite
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `test` ..... soubory použité při testování
  - `sources` ..... zdrojové kódy použité při testování
  - `applets` ..... applety použité v při testování
- `thesis.pdf` ..... text práce ve formátu PDF
- `readme.txt` ..... popis obsahu CD

Příloha C

Autentikace

C.1 INITIALIZE UPDATE

Tabulka C.1: APDU příkaz INITIALIZE UPDATE

Bajt	**Hodnota**	**Popis**
CLA	0x80 – 0x83 nebo 0xC0 – 0xCF	Třída příkazů a výběr logického kanálu
INS	0x50	INITIALIZE UPDATE
P1	xx	Verze klíče, kterou použije bezpečnostní doména
P2	00	Musí se rovnat 0x00
Lc	0x08	Délka Host Challenge
Data	xx xx ...	Host Challenge, generuje se off-card entitou a v ramci sessiony musí byt unikatní
Le	00	Musí se rovnat 0x00

C.2 EXTERNAL AUTHENTICATE

Tabulka C.2: APDU příkaz EXTERNAL AUTHENTICATE

Bajt	**Hodnota**	**Popis**
CLA	0x84 – 0x87 nebo 0xE0 – 0xEF	Třída příkazů, výběr logického kanálu, identifikace zabezpečené zprávy
INS	0x50	EXTERNAL AUTHENTICATE
P1	xx	Víz. C.3
P2	00	Musí se rovnat 0x00
Lc	0x10	Délka Host Cryptogram a C-MAC
Data	xx xx ...	Host Cryptogram a C-MAC
Le		Není

Tabulka C.3: Hodnoty P1 EXTERNAL AUTHENTICATE

P1	**Security Level**	**Protokol**	
0x19	C-DECRYPTION + C-MAC + R-MAC		SCP02
0x17	C-MAC+R-MAC		
0x10	R-MAC		
0x03	C-DECRYPTION+C-MAC	SCP01	
0x01	C-MAC		
0x00	NO_SECURITY_LEVEL		