VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MODERNÍ PŘÍSTUPOVÝ SYSTÉM
MODERN ACCESS CONTROL SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Martin Vomáčka

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Lukáš Malina, Ph.D.

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Martin Vomáčka ***ID:*** 125339
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Moderní přístupový systém

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je návrh a implementace bezpečného přístupového systému pro chráněné fyzické a logické prostory. Navržený systém využije vybraných čipových karet, moderních kryptografických metod a síťových komponentů. Výsledné řešení by mělo dbát především na celkovou bezpečnost (bezpečnost autentizačního protokolu, zabezpečení kom. kanálů) a také na efektivitu (doba odezvy při autentizaci, cena komponentů). Součástí systému bude i návrh podsystému pro sledování autentizovaných uživatelů a vyhodnocování jejich profilů. Navržený systém bude implementován na vybrané platformě (aplikace ověřovatele), čipové kartě (aplikace uživatele) a experimentálně ověřen.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, William. Cryptography and Network Security. 4th edition. [s.l.] : [s.n.], 2006. 592 s. ISBN 0131873164.

[2] MENEZES, Alfred, VAN OORSCHOT, Paul, VANSTONE, Scott. Handbook of applied cryptography. Boca Raton : CRC Press, 1997. 780 s. ISBN 0849385237.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 25.5.2016

Vedoucí práce: Ing. Lukáš Malina, Ph.D.
Konzultant diplomové práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Diplomová práce se zaměřuje na návrh schématu přístupového systému s autentizací uživatelů pomocí smart karet. V první části práce jsou popsány dostupné druhy identifikačních předmětů pro autentizaci uživatelů a také druhy čtecích zařízení. Druhá část se potom podrobně věnuje čipovým kartám, popisuje druhy čipových karet a také vnitřní strukturu a princip komunikace těchto karet se čtecími zařízeními. Tato část se věnuje primárně popisu java karet. Třetí část se zabývá kryptografií na platformě Java Card a věnuje se hlavně eliptickým křivkám. V kapitole čtvrté je představen protokol PACE, jsou rozebrány jeho dílčí části a způsob, jakým je tento protokol aplikován na čipové karty. Pátá část se věnuje detailnímu popisu navrženého přístupového systému, jeho dílčích částí a popisu funkčnosti a ovládání vytvořených aplikací.

KLÍČOVÁ SLOVA

Smart karta, java karta, šifrování, autentizace, hash, řízení přístupu, Java, PACE, AES, ECC, MAC

ABSTRACT

The thesis describes the design of scheme for access system with user authentication via smart cards. The first chapter explains various types of identification items used for authentication of users and different types of readers and terminals, followed by chapter 2 with a deeper insight on smart cards with focus on their types, what internal structure and principle of communication with card readers are used, etc. with primary focus on Java cards. The third chapter describes Java Card cryptography - especially elliptic curve cryptography used on this platform. The fourth part focuses on PACE protocol with subsections dedicated to the individual parts of the protocol and its applicability to smart cards environment. Chapter 5 explains the proposed design of the authentication scheme elaborated in the thesis, including a detailed description of specific parts, their funcionality and exemplary usage in the created applications.

KEYWORDS

Smart Card, Java Card, Encryption, Authentication, Hash, Access control, Java, PACE, AES, ECC, MAC

VOMÁČKA, Martin *Moderní přístupový systém*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 70 s. Vedoucí práce byl Ing. Lukáš Malina, PhD.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Moderní přístupový systém“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce 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 uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Lukáši Malinovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Čipové karty a další předměty, sloužící k digitálnímu prokázání identity nebo autentizaci, jsou v současnosti neodmyslitelným prvkem používaným takřka denně. Od platebních karet, předplacených karet, mobilních SIM karet až po ty identifikační. Všude zde je potřeba zajistit ochranu osobních údajů a opatřit tyto předměty dostatečným zabezpečením proti jejich zneužití nebo úniku těchto citlivých informací. Aby toho mohlo být dosaženo jsou v drtivé většině případů komunikace a samotná data uchovaná na kartách šifrována. Aby bylo šifrování spolehlivé a rychlé, disponují karty vlastními implementovanými kryptografickými prostředky.

V první kapitole se stručně seznámíme s nejrůznějšími druhy přístupových systémů. Protože přístupový systém lze rozdělit na identifikační předměty pro identifikaci a na samotné systémy, zajišťující ověření těchto předmětů a udělování přístupu, probereme pár základních typů identifikačních předmětů a druhy a typy čtecích zařízení. Současným trendem je nástup bezkontaktních technologií a na tento fakt se práce hlavně zaměřuje - využití bezkontaktních čipových karet a čtecích zařízení.

Druhá kapitola se podrobněji věnuje popisu čipových karet. Je zde rozebrán popis hardwarového vybavení čipových karet a princip komunikace se čtecími zařízeními s detailním popisem komunikačního protokolu. Dále je také naznačena struktura a funkčnost softwarových dispozic čipových karet. V této práci jsem pracoval pouze s Java kartami, proto je samotný popis softwarových dispozic zaměřen hlavně na platformu JavaCard, včetně stručného popisu tvorby a instalace aplikací na Java karty.

Kapitola třetí je zaměřena na kryptografické dispozice platformy Java Card. Stručně popisuje historii a vývoj samotné platformy a také se věnuje způsobu získání podporovaných kryptografických prostředků na čipových kartách. Dále je kapitola zaměřena hlavně na kryptografii nad eliptickými křivkami a to, jakým způsobem je tento zakomponována do platformy JavaCard. Věnuje se také stručné analýze eliptických křivek, které karty podporují a časové analýze operací nad křivkami.

Čtvrtá kapitola sezabývá teorií hlavní části této práce, protokolu PACE. Na začátku je představen samotný protokol a jeho parametry a poté jsou do detailů rozebrány jednotlivé části protokolu a popis toho, jak byl tento protokol aplikován na platformu Java Card.

Pátá kapitola je zaměřena na konkrétní návrh celého přistupového systému a jeho implementaci. Popisuje zakomponování protokolu PACE a dalších bezpečnostních opatření do jednoho systému. Jednotlivé části přístupového systému jsou dopodrobna popsány - navržený applet pro čipové karty, terminálové aplikace a také databázový server. Kapitola obsahuje popis GUI jednotlivých aplikací a jejich ovládání. Pozornost je věnována také časové a bezpečnostní analýze navrženého systému.

1 PŘÍSTUPOVÉ SYSTÉMY

Přístupové systémy slouží všude tam, kde je potřeba nějakým způsobem omezit, regulovat nebo řídit přístup uživatelů k určitým prostředkům a také zamezit přístup uživatelům nepovolaným. Těmito prostředky můžeme rozumět například nemovité objekty (budovy, firmy, areály), informace (data uvnitř firemní sítě, soukromé soubory na disku), služby (menza, používání firemní kopírky, emailová schránka, VoIP telefon). Kromě řízení přístupu se dá systém také využít pro monitorování uživatelů, jejich pohyb nebo frekvenci využívání daných služeb, dále také statistické zpracování přístupů pro případné optimalizace služeb nebo jiný druh chování v důsledku těchto statistických údajů.

Jednotlivé součásti přístupových systému se dají z obecného hlediska rozdělit na tři kategorie. První kategorií jsou centrální servery nebo jiná zařízení, které řídí celý přístupový systém. Starají se o chod celého systému, udělování nebo případné zamítání přístupů jednotlivým uživatelům, zaznamenávání statistik a ovládání komponent, ke kterým je tímto serverem přístup zajišťován. Mohou to být jednoduché mikrokontroléry připojené na pár koncových terminálů nebo naopak velké servery zajišťující přístup k objektům, informacím i službám pro celou firmu zároveň. Vše záleží na konkrétních podmínkách.

Druhou kategorií jsou koncové terminály, pomocí kterých uživatelé provádějí svou autentizaci a pomocí kterých přistupují k prostředkům. Terminálem je rozuměno jakékoliv zařízení, prostřednictvím kterého uživatel prokazuje svou identitu centrálnímu serveru. Mohou to být osobní počítače, telefony, snímače otisků, čtečky přenosných karet, klávesové zámky atd.

Třetí kategorií jsou pak prostředky, kterými uživatelé svou identitu prokazují. Jednoduchá přístupová hesla, uživatelské přihlašovací údaje (login a heslo), biometrické údaje (otisk prstu, obraz oka, hlas) nebo předměty prokazující uživatelovu identitu (průkaz, RFID tag, čipová karta).

Protože je tato práce zaměřená hlavně na čipové karty a autentizaci pomocí nich, budu se v následujících podkapitolách věnovat hlavně dostupným prostředkům z oblasti čtecích zařízení a typů přístupových předmětů (čipové karty, RFID tagy, terminály, čtecí zařízení apod.).

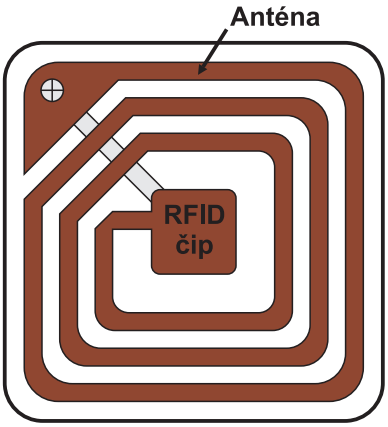
1.1 Typy identifikačních předmětů

Aby mohl uživatel přístupovému systému prokázat svou identitu a získat tak přístup, musí se při autentizaci prokázat předmětem, který nese údaje o tomto uživateli a zároveň je tento předmět věrohodný. Nejčastějším zástupcem takového předmětu je

čipová karta, RFID tag nebo také v dnešní době stále oblíbenější využití chytrých telefonů vybavených NFC technologií ve spojení s adekvátní autentizační aplikací v telefonu.

1.1.1 RFID tagy

RFID tagy jsou o jednoduchá zařízení fungující jako bezkontaktní identifikátory pracující na radiových frekvencích. Ve své podstatě se jedná pouze o jednoduchý čip s drobnou pamětí, nesoucí svůj unikátní identifikátor a anténu, pomocí které komunikuje, jak je možné vidět na obrázku č. 1.1. Tagy je možné rozdělit dle své funkčnosti na aktivní a pasivní, přičemž aktivní mají od těch pasivních navíc i vlastní baterii – jedná se o tzv. TTF tagy[1]. Pasivní čipy komunikují pouze tehdy, dostanou-li se do blízkosti čtecího zařízení – jedná se o tzv. RTF tagy[2]. Obě varianty mají své výhody a nevýhody. Výhodou aktivních tagů je jejich mnohonásobně vyšší dosah (desítky až stovky metrů), nevýhodou však životnost a odolnost (kvůli přítomnosti baterie). Výhodou pasivních tagů je jejich nízká cena, vysoká odolnost a životnost, bohužel jejich dosah se pohybuje maximálně v jednotkách metrů [10]. Pasivní tagy jsou velice rozšířené a považují se za moderního nástupce čárových kódů pro identifikaci produktů. Největší výhodu RFID tagů je fakt, že pomocí čtecích zařízení je možné načítat velké množství tagů zároveň a také možnost zápisu či změny informací, kterou nesou.



Obr. 1.1: Vnitřní struktura RFID tagu

[1]TTF - tag talks first. Tagy samy o sobě vysílají své údaje do okolí a mohou tak zahajovat komunikaci se čtečkou

[2]RTF - reader talks first. Tagy pouze odpovídají na komunikace od čtečky, když se dostanou do jejich blízkosti.

1.1.2 Čipové karty

Svým vzhledem jednoduchá plastová karta, která však v sobě nese zabudovaný mikročip, který umí uchovávat a zpracovávat dodatečné informace o držiteli karty. Čipová karta umí zapisovat a zpracovávat transakce, které byly/budou s kartou provedeny a nad těmito transakcemi mohou být prováděny další operace bez nutnosti zpracování operací mimo kartu – šifrování, digitální podpisy, apod. Čipovým kartám se často říká Smart karty z důvodu, že mohou bezpečně uchovávat a šifrovat data na ně přenesená a je těžké je falšovat nebo padělat. Jsou používány jako telefonní, zdravotní nebo bankovní karty, aktivační karty pro placené televizní kanály, GSM apod. Více o čipových kartách pojednává kapitola č. 2.

1.2 Typy čtecích zařízení

Tato zařízení mají za úkol spolehlivě a pokud možno rychle načíst informace z identifikačního předmětu. Čtecí zařízeni (dále jen čtečky) je možné rozdělit podle dvou parametrů. Prvním parametrem je, zda při čtení je potřebný fyzický kontakt s identifikačním předmětem a nebo je možné identifikaci provést bezkontaktně. Druhým parametrem je, zda čtečka obstarává pouhé načtení informace a tuto informaci pak předává dál nějakému nadřazenému systému, který ji následně vyhodnotí a nebo samotné čtecí zařízení získané informace přímo vyhodnocuje – v takovém případě hovoříme o terminálu.

1.2.1 Čtečky a terminály

Úkol čtečky je jednoduchý – v případě, zda se v jejím dosahu nebo čtecím slotu objeví identifikační předmět, načte informace o tomto předmětu a předá je k dalšímu zpracování jinému systému. Samozřejmě taková čtečka může naopak sloužit i k zápisu informací zpět na identifikační předmět, ale i tak je vždy čtečka chápána pouze jako technologický prostředník mezi dvěma zařízeními (systémy), který obstarává komunikaci. Může se například jednat o RFID bránu, která neustále ve svém prostoru vyhledává RFID tagy a pokud nějaký načte, tak informaci o něm předá například sériovou linkou nebo pomocí klasické TCP/IP sítě dále ke zpracování.

Druhým příkladem takové čtečky je například USB čtečka čipových karet. Obsahuje pouze kontaktní/bezkontaktní rozhraní (nebo obojí) a jejím výstupem je sériová USB linka pro připojení k počítači – jako například čtečka karet použitá v tomto projektu HID OMNIKEY 5321v2 viz. obr. č. 1.2, která má duální rozhraní pro obojí kontaktní a bezkontaktní čipové karty.

Obr. 1.2: USB čtečka karet s duálním rozhraním HID OMNIKEY 5321v2

Pokud mluvíme o terminálu, tak popisujeme komplexní systém, který neprovádí pouhé čtení/zápis a předávání informací. Takový terminál už s načtenými informacemi sám pracuje a vyhodnocuje je. Terminály jsou nejčastěji používány jako vestavěná zařízení, kde je vyžadována komplexní funkčnost takového systému na jednom místě. Typickým příkladem terminálu je bankomat nebo platební terminál v obchodě, zobrazený na obrázku č. 1.3. Obsahuje jak čtecí zařízení na čipové karty, tak svůj vlastní vestavěný systém, který ovládá klávesnici bankomatu, mechanismus pro počítání a výdej peněz, displej, zobrazování grafiky, tisk stvrzenek a další věci. Jiným příkladem terminálu může být přenosná čtečka RFID tagů v nákupním středisku, kterou zaměstnanec může skenovat jednotlivé zboží a případně provádět určité změny dle potřeb. Takový příruční terminál bude s nejvyšší pravděpodobností obsahovat nějaký displej a klávesnici pro zadávání informací.

Obr. 1.3: Platební terminál PAX S80

1.2.2 Kontaktní a bezkontaktní čtečky

Kontaktní čtečky vyžadují fyzický kontakt s předmětem, ze kterého čte informace. V realitě to znamená vložení nebo přiložení identifikačního předmětu na kontakty nebo konektor čtečky. Od tohoto principu se však v dnešní době pomalu ustupuje, i když výhodou kontaktních čteček jsou vyšší přenosové rychlosti – napájení čipu, generování hodinového signálu a samotný přenos dat je realizován pomocí k tomu určených kontaktů. Také je sníženo nebezpečí odposlechnutí komunikace nějakým útočníkem. Nevýhodou však je právě samotné používání kontaktních karet, kdy dochází k opotřebení kontaktů jak ve čtečce, tak na samotných kartách. Také je nutné dodržovat správnou orientaci karty při vkládání do čtečky.

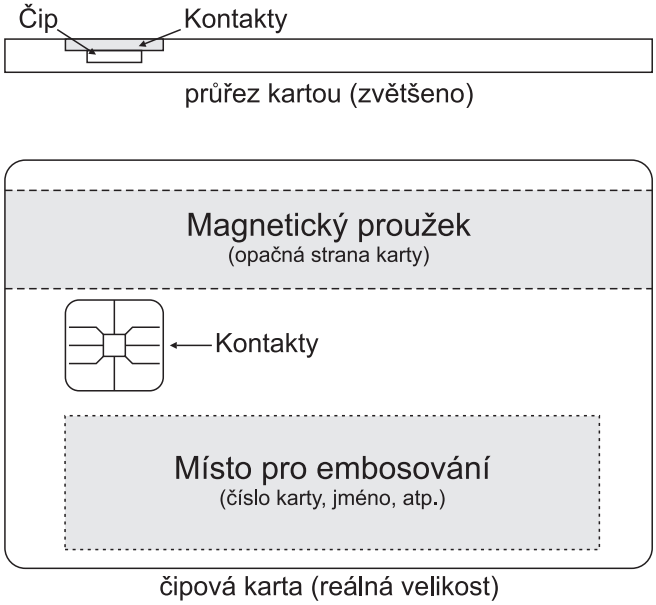
U bezkontaktních čteček tento problém s fyzickým namáháním odpadá. Sice dochází k omezení přenosových rychlostí z důvodu napájení pomocí elektromagnetické indukce přes anténu, ale protože zpravidla nedochází k přenosům velkých objemů dat, je tato nevýhoda zanedbatelná. Jediné co je potřeba u bezkontaktních čteček dodržovat, je určitá maximální vzdálenost od identifikačního předmětu, aby mohla být karta napájena a mohla probíhat komunikace.

2 ČIPOVÉ KARTY

Čipové karty jsou v dnešní době již nedílnou součástí každodenního života. Jejich nejznámější využití je v oblasti bankovnictví, dále pak také různé předplatné systémy typu městská hromadná doprava, vstupní permanentky a v neposlední řadě také jako identifikační průkazy a přístupové karty. Čipové karty na rozdíl od magnetických karet obsahují vlastní mikroprocesor a mohou informace nejen uschovávat, ale také s nimi provádět různé operace přímo na kartě. Co se týká velikosti čipových karet, nejčastěji je možné se setkat s kartami typu ID-1 o rozměrech 85,60×53,98 mm podle standardu ISO/IEC 7816-1. Druhým také velice známým typem je ID-000 o rozměrech je 25×15 mm, pro většinu lidí však známý jako telefonní SIM karta. Samotná struktura čipové karty je pak dále upřesněna ve standardu ISO/IEC7816-2, který upřesňuje rozmístění jednotlivých komponent na kartě a některé fyzikální vlastnosti. Dodržení rozměrů je důležité hlavně pro kontaktní karty, co se týká těch bezkontaktních, tam jsou tvary a rozměry karet libovolné [15].

Volba rozměru kontaktních karet typu ID-1 má svůj důvod, a tím je zpětná kompatibilita se staršími magnetickými kartami. Většinou tak bývá čipová karta kombinována s magnetickým proužkem pro případ použití ve starších magnetických systémech. Na obrázku č. 2.1 je ilustrováno umístění jednotlivých částí čipové karty.

V této kapitole se zaměřím na rozdělení čipových karet, hardwarové možnosti, způsob komunikace a komunikační protokol a také strukturu operačních systémů, které na kartách zajišťuji samotnou funkcionalitu aplikací instalovaných na kartě.



Obr. 2.1: Umístění jednotlivých prvků čipové karty

2.1 Rozdělení čipových karet

Čipové karty je v širším hledisku možné dělit podle dvou parametrů. Zaprvé je můžeme dělit podle toho, jestli se jedná o pouhé paměťové médium nebo má karta kromě paměti také vlastní mikroprocesor. Výraz čipové karty je tak obecně používán pro všechny druhy karet, ale většinou pouze ty mikroprocesorové jsou označovány jako SmartCards neboli chytré karty. Zadruhé se dají karty dělit podle způsobu komunikace s terminálem na karty kontaktní a bezkontaktní.

2.1.1 Paměťové a mikroprocesorové karty

Prvním druhem karet, které se začaly vyrábět ve velkém, byly karty paměťové. Paměťové karty nelze až tak přímo považovat za „chytré“, protože neobsahují mikroprocesor, ale jsou vybaveny pouze základní pamětí ROM a EEPROM o velikostech většinou v jednotkách kilobajtů. Dále se na kartě nachází také několik základních logických obvodů, které zajišťují práci s pamětí pomocí předprogramovaných jednoduchých instrukcí.

Tyto karty disponují úložištěm v řádech jednotek kilobajtů a protože nemají žádný procesor, není možné je po výrobě přeprogramovat pro jiné nebo nové využití. Nejčastěji dochází k operacím typu čtení hodnot, případně jejich inkrementace či dekrementace. Takovéto karty jsou často využívány jako předplatné karty na omezený počet použití, třeba jako předplacené nebo reklamní kreditové SIM karty telefonních operátorů. Po vyčerpání určitého obnosu (např. určitý počet SMS nebo určitý počet provolaných minut) buď přestanou fungovat a uživatel ji může vyhodit bez nutnosti návratu společnosti, která mu kartu vydala (tzv. jednorázové karty). A nebo je možné některé karty opět dobít a pokračovat tím v jejich využívání.

Zabezpečení paměťových karet spočívá většinou v blokování přístupu k určitým oblastem paměti pomocí bezpečnostních klíčů nebo možnosti provádět pouze povolené operace s pamětí. Pokud se tedy jedná například o nějaký systém přednabitého kreditu, uživatel může hodnotu kreditu pouze dekrementovat, ale nikoliv inkrementovat – k této funkci má (v případě dobitelné karty) přístup pouze vydavatel karty pomocí bezpečnostních klíčů. Nevýhodou jednorázových paměťových karet je však jejich jednoduchost a tak je snadnější takové karty padělat, protože neobsahují žádné zabezpečené oblasti paměti.

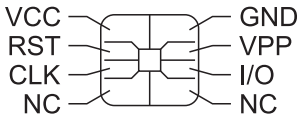
Jak název napovídá, mikroprocesorové karty disponují procesorem. Hlavním přínosem je jejich programovatelnost, vyšší možnosti zabezpečení a jejich multifunkčnost. U těchto karet nekomunikuje terminál přímo s paměťovým úložištěm karty, ale pouze s mikroprocesorem, který přístup k informacím na kartě zprostředkovává. Tím je možné zajistit vyšší zabezpečení informací na kartě pomocí hesel, kryptografie a

také tím, že přístup k paměti není možný jinak než prostřednictvím mikroprocesoru. Tyto karty jsou velice všestranné, mohou být optimalizovány pro jednu specifickou aplikaci nebo je možné na jedné kartě používat více různých aplikací [1]. Limity v tomto případě udávají pouze hardwarové vlastnosti karty (velikost paměti, rychlost procesoru, dostupné kryptografické prostředky, atd.), případně pak cena.

Oproti paměťové kartě, ta mikroprocesorová má kromě základní ROM a EEPROM paměti navíc také paměť RAM, samozřejmě svůj vlastní procesor a často také přídavný kryptografický koprocesor. Nejčastějšími parametry karet bývá 8-bitový procesor, 16kb ROM, 16kb EEPROM, 4kb RAM a kryptografický koprocesor pro urychlení operací souvisejících se zabezpečením.

2.1.2 Kontaktní a bezkontaktní karty

Kontaktní karty musí být pro komunikaci s terminálem vloženy do čtečky, přičemž je důležité dodržet správnou orientaci karty při vkládání do čtecího zařízení. Komunikace potom probíhá přes kontakty na kartě pomocí 8-pinového rozhraní – obrázek č. 2.2 popisuje jednotlivé kontakty tohoto rozhraní. Tyto karty bývají zpravidla levnější než bezkontaktní, ale na druhou stranu se používáním opotřebovávají jejich kontakty. Časem může dojit k poškození a tím znemožnění další komunikace se čtecím zařízením. Pochopitelně je také důležité u kontaktních karet dodržet umístění kontaktů [11].



VCC	napájení
GND	uzemnění
RST	signál resetování komunikace
VPP	programování EEPROM
CLK	hodinový signál
I/O	sériové rozhraní
NC	nevyužito

Obr. 2.2: Rozmístění a význam kontaktů čipové karty

U karet bezkontaktních odpadá problém se správnou orientací karty a jejich používání je tak velice jednoduché a nachází uplatnění v případech, kdy je potřeba provádět rychlé operace s kartou – bezkontaktní platby, turnikety, přístupové systémy. U bezkontaktních karet je napájení karty řešeno pomocí elektromagnetické indukce a pro správnou funkčnost a komunikaci s kartou je potřeba dodržovat určité vzdálenosti mezi kartou a čtečkou. Protože jsou veškeré součásti karty zatavené uvnitř samotné karty, nedochází jejím používáním k žádnému opotřebení a kartu je tak možné používat bez obav o její životnost (samozřejmě ne v případě fyzického poškození karty). Protože komunikace karty se čtečkou většinou probíhá jen v krátkém

okamžiku, může se stát, že komunikace neproběhne celá a přeruší se. Bezkontaktní karty bývají zpravidla dražší než kontaktní a v současnosti se velmi často setkáváme s kartami hybridními, které jsou jak kontaktní, tak bezkontaktní.

2.1.3 Komunikační protokol

Čipové karty a čtecí zařízení používají pro vzájemnou komunikaci svůj vlastní komunikační protokol, který přesně popisuje standard ISO/IEC 7816-4. Komunikační protokol lze rozdělit na 3 vrstvy [2]. První a nejvyšší je aplikační vrstva, kde komunikace probíhá pomocí příkazů, které jsou známy jen aplikaci terminálu a k ní odpovídající aplikaci na kartě. Druhá vrstva se skládá ze zpráv APDU, jejichž formát je nezávislý na aplikacích, ale jejich obsah a význam ano (tzn. lze jednu zprávu APDU zpracovat každou aplikací úplně odlišně). Poslední nejnižší vrstva již zajišťuje samotný přenos informací mezi kartou a čtečkou a skládá se z TPDU zpráv. Tato vrstva pracuje ve dvou režimech T=0 a T=1. V případě T=0 probíhá komunikace po bajtech, což je na jednu stranu sice jednoduché, ale z pohledu bezpečnosti snadno zneužitelné, protože požadavek a odpověď z karty jsou přenášeny ve dvou samostatných výměnách. Režim T=1 komunikuje po blocích bajtů a dokáže požadavek a odpověď z karty přenést v jedné výměně dat, čímž lépe odděluje aplikační od transportní vrstvy a komunikace již není tak snadné rozluštit.

Komunikace mezi čtečkou a kartou lze rozdělit do několika fází. První fází je vložení karty do čtečky (nebo přiložení ke čtečce, jedná-li se o bezkontaktní kartu). Po připojení napájení na kartu proběhne resetování adresového čítače mikroprocesoru a tím jeho počáteční inicializace. Odpovědí karty na tento reset je zpráva ATR, která nese základní informace o kartě a je také signálem k tomu, že karta je připravena komunikovat. Pomocí zprávy ATR je možné provést základní identifikaci čipové karty, jako například výrobce, hardwarové vlastnosti či typ operačního systému na kartě a také je pomocí ní identifikován protokol přenosové vrstvy, se kterým karta pracuje (T=0 nebo T=1)[1]. Následně probíhá vzájemná komunikace pomocí APDU zpráv a ukončení komunikace je provedeno jednoduše vyjmutím karty ze čtečky.

APDU – struktura a syntaxe

Zprávy APDU slouží pro vzájemnou výměnu informací a přenosu dat mezi čtečkou a kartou a je možné je rozdělit na dvě kategorie:

- **Command APDU** (dále jen **C-APDU**), jsou řídicí zprávy, které jsou vysílány čtečkou karet (terminálem) a říkají čipové kartě jakou operaci má vykonat a jakým způsobem reagovat

[1]Pro účely snadnější identifikace karty pomocí ATR existuje webová aplikace pro parsování ATR odpovědí: `https://smartcard-atr.appspot.com/`

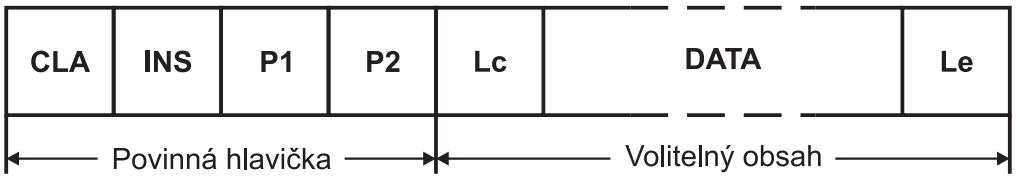
- **Response APDU** (dále jen **R-APDU**), jsou zprávy odpovědí, které jsou vysílány čipovou kartou a jedná se o potvrzovací zprávy na přijaté řídicí příkazy

Celá komunikace probíhá tak, že terminál vyšle C-APDU, na který karta odpoví R-APDU, tedy komunikace probíhá v párech a vždy začíná dotazem C-APDU, nikdy komunikaci nezačíná čipová karta. Formát a syntaxe jednotlivých zpráv se liší, proto je podrobně rozeberu v jednotlivých podkapitolách.

Command APDU

Řídicí zpráva s skládá ze dvou částí – povinná hlavička a volitelný obsah. Struktura zprávy je znázorněná na obrázku č. 2.3. Povinná hlavička se sestává ze 4 bajtů a je po bajtech rozdělená na jednotlivé části. Pokud jsou ve zprávě přenášena další data, může pak zpráva obsahovat kromě samotných dat ještě další dvě přídavná pole[2]. Význam jednotlivých prvků je následující:

- **CLA** – udává kategorii prováděného příkazu
- **INS** – udává příkaz, který se má provést
- **P1, P2** – přídavné parametry příkazu
- **Lc** – specifikuje délku odesílaných dat, může být dlouhé až 3 bajty
- **DATA** – přenášená data na kartu o maximální délce **Lc**
- **Le** – specifikuje délku dat očekávaných v odpovědi od karty



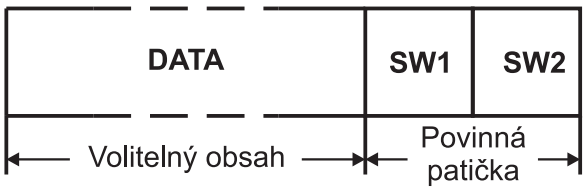
Obr. 2.3: Struktura Command APDU

Response APDU

Zpráva odpovědi, podobně jako zpráva řídicí se opět skládá z povinné a nepovinné části. Povinnou částí je vždy dvojice bajtů **SW1** a **SW2**, které tvoří patičku zprávy a nepovinnou částí jsou data přenášená z karty na terminál. Struktura zprávy je znázorněna na obrázku č. 2.4. Nepovinná datová část je přenášena pouze v případě, že si přenos dat terminál vyžádal a data přenášená z karty by měla mít správnou délku, kterou specifikovalo pole **Le** v řídicí zprávě. Dvojice bajtů SW1 a SW2 slouží buď jako odpověď na prováděný příkaz nebo v případě nějaké chyby také jako informace o chybě. Ve své podstatě se jedná o návratovou hodnotu při volání funkce, která

[2]Kompletní seznam řídících zpráv a jejich významů je k dispozici na: `http://web.archive.org/web/20090630004017/http://cheef.ru/docs/HowTo/APDU.info`

v případě potřeby umí informovat o chybách nebo přenést zpět nějaká data. Například odpověď „0x9000“ znamená úspěšné provedení požadované operace – příjem dat, vykonání funkce na kartě a další[3].



Obr. 2.4: Struktura Response APDU

Možné varianty komunikace

Jelikož C-APDU i R-APDU mohou obsahovat volitelné položky, dá se komunikace mezi kartou a terminálem rozdělit na 4 případy:

1. V prvním případě nedochází k žádnému přenosu dat z terminálu na kartu ani naopak. Z terminálu je tak odeslána pouze 4 bajtová hlavička a v odpovědi přijde jen 2 bajtová patička.
2. V druhém případě probíhá přenos dat z terminálu na kartu, ale nejsou očekávána žádná data z karty na terminál. C-APDU se tak bude skládat pouze z hlavičky, pole Lc a odesílaných dat. V odpovědi R-APDU pak přijde opět pouze jen 2 bajtová patička.
3. Ve třetím případě je terminálem vyžadován přenos nějakých dat z karty. C-APDU bude obsahovat pouze hlavičku a pole Le, udávající očekávanou délku dat v odpovědi. Tuto délku je možné kontrolovat, ale není to povinné, záleží pouze na programátorovi dané aplikace. V odpovědi R-APDU pak budou jako první odeslána data a na konci opět 2 bajtová patička.
4. V posledním případě jsou data přenášena oběma směry, tudíž C-APDU i R-APDU budou obsahovat všechna možná pole.

2.2 Struktura operačních systémů čipových karet

Operačním systémem čipových karet lze rozumět určitou sbírku funkcí a instrukcí, kterých může následně uživatel využívat při tvorbě svých aplikací. Většinou se jedná

[3]Kompletní seznam odpovědí a jejich významů je k dispozici na: `https://www.eftlab.com.au/index.php/site-map/knowledge-base/118-apdu-response-list`

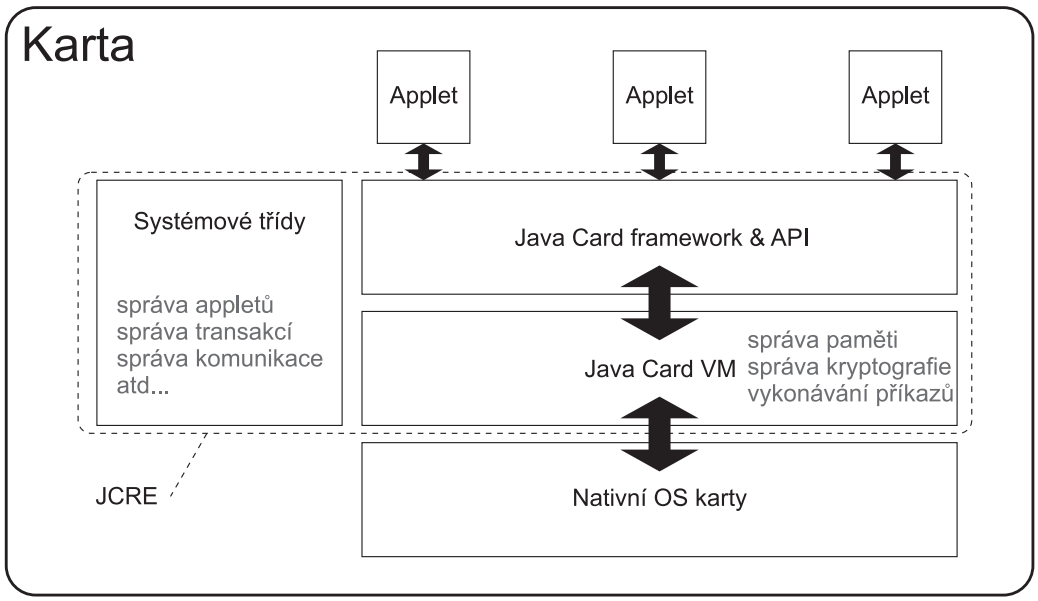
o různou práci se soubory, funkce pro zpracování APDU, kryptografické funkce v závislosti na dostupných prostředcích kryptografického procesoru. Tento základní operační systém je součástí každé karty, už pomocí něj by bylo možné na kartě provozovat aplikace. Ty by však musely být psány ve strojovém jazyce daného mikroprocesoru na kartě. Aby bylo možné využívat vyšších programovacích jazyků, jako je například Java, C, C#, je potřeba, aby byla v kartě nástavba základního operačního systému, která by takový vyšší programovací jazyk podporovala. Potom je možné psát aplikace pro čipovou kartu v těchto jazycích, přičemž je možné využívat stejnou syntaxi daného programovacího jazyka.

Protože karty disponují omezenými paměťovými a výpočetními prostředky, není možné využívat veškerých knihoven a funkcích dostupných ve standardních specifikacích daného programovacího jazyka. Proto jsou na čipových kartách k dispozici pouze základní datové typy, základní funkce pro práci s datovými poli a také široká nabídka šifrovacích knihoven a kryptografických mechanismů. To, jaké mechanismy jsou k dispozici vždy závisí na vlastnostech čipové karty. Některé čipové karty mohou podporovat například jen symetrické šifrování, jiné naopak zase pouze asymetrické. Možností je hodně a proto je vždy dobré vědět, s jakým typem karty uživatel pracuje, aby mohl podle toho volit nejvhodnější metody. Protože v této práci pracuji hlavně s Java Card, bude následující popis struktury OS zaměřen na programovací jazyk Java.

2.2.1 Java Card Runtime Environment

Java Card Runtime Environment (dále jen JCRE) v sobě zapouzdřuje několik různých komponent pro úspěšné používání aplikací napsaných v Javě na čipové kartě. Poskytuje nástroje pro komunikaci s čtečkou karet, tedy zpracování APDU, dále obsahuje instalátor appletů, pro jejich úspěšné nasazení na kartu a rozhraní pro nativní (základní) operační systém karty. Další důležitou součástí jsou systémové třídy a balíčky programovacího jazyka Java, kterých jednotlivé applety využívají. Protože není možné, aby na kartě byly k dispozici všechny dostupné třídy a funkce jazyka Java, jsou dostupné pouze některé třídy, objekty a datové typy tak, aby bylo možné co nejvíce využít potenciálů čipových karet a zároveň poskytnout uživateli dostatek volného místa na kartě pro realizaci jeho vlastních appletů. Také obsahuje jednoduchý interpretr JCVM, který zajišťuje překlad a vykonávání jednotlivých příkazů do strojového jazyka nativního systému karty. Také řídí alokování paměti a kryptografii. Na obrázku č. 2.5 je znázorněna struktura JCRE na čipové kartě.

Protože karta musí umět uchovat informace i po odpojení napájení, jsou veškeré objekty a perzistentní data uživatele (například kryptografické klíče) ukládány do paměti EEPROM. Při instalaci appletu na kartu vytvoří JCRE instance appletů



Obr. 2.5: Struktura JCRE na čipové kartě

a applety si pak vytvoří potřebné objekty v perzistentní paměti pro uložení dat. Tento proces probíhá vždy jen jednou při instalaci appletu na kartu. Pokud pak chceme s appletem pracovat, provádění příkazů už má na starost JVCM. Když dojde k odpojení od napájení, perzistentní data na kartě zůstanou a činnost JVCM je pouze pozastavena. To však neznamená, že po obnovení napájení dokáže JVCM pokračovat dále ve své činnosti. Naopak dojde k resetování jeho stavu a tím znovunačtení dat z perzistentní paměti, aby byl připraven k chodu. Pokud došlo k výpadku napájení v průběhu vykonávání nějaké operace, JCRE dokáže zařídit potřebný úklid paměti z nedokončené transakce a návratu do konzistentního stavu před jejím provedením.

2.2.2 Java Card API

Obsahuje kolekci balíčků a tříd, které je možné využívat pro tvorbu a fungování vytvořených appletů. Jedná se vlastně o omezenou verzi Javy, kdy máme k dispozici určité balíčky pro práci s datovými strukturami, pro komunikaci a zpracování APDU, kryptografii a šifrování. Aktuální verzí API je 3.0.5, ale je nutné podotknout, že pro správnou funkčnost a možnost využití jeho funkcí je nutné vlastnit také kartu, která tuto verzi API podporuje. Bohužel co se týká samotných karet, nefunguje zde zpětná kompatibilita a je nutné pro nasazení appletů programovat v tom API, které karta podporuje – tím je namysli to, že karta podporující JCAPI 2.2.1 nedokáže pracovat s appletem napsaným v API verze 2.2.2 a vyšší. To, jakou verzi API karta podporuje, se uživatel dozví od distributora karty nebo je možné tuto informaci vyčíst přímo z karty, kde je tato informace taktéž uložena.

Základním stavebním kamenem API je balíček javacard.framework, zajišťující chod appletů, jejich komunikaci s JCRE a také zpracování APDU a v neposlední řadě také práci s pamětí. Dalším důležitým balíčkem je javacard.security, zajišťující rozhraní pro kryptografické funkce, hlavně pak pro tvorbu a uchování šifrovacích klíčů, generování náhodných čísel apod. Posledním důležitým balíčkem je javacardx.crypto, který poskytuje rozšířené kryptografické funkce a využívá dostupných možností vestavěných kryptografických koprocesorů, kterými jsou karty vybaveny.

2.2.3 Tvorba appletů pro Java Card

Aby bylo možné úspěšně vytvořit fungující applet, je potřeba dodržet určitá pravidla. Prvním a pochopitelným je správná verze JCAPI tak, aby bylo možné applet na kartu nainstalovat a provozovat ho. Samotná třída appletu musí rozšiřovat třídu javacard.framework.Applet. Protože je možné, aby na jedné čipové kartě bylo nainstalováno více appletů, druhou důležitou věcí je přiřadit appletu identifikátor, který musí být v rámci jedné čipové karty unikátní – jako zkratka se používá AID. Při instalaci appletu se tento identifikátor kontroluje a pokud už na kartě applet s daným identifikátorem je, k instalaci nedojde. Identifikátor také slouží k výběru aktivního appletu na kartě.

Tvar AID specifikuje standard ISO/IEC 7816-5 a udává, že délka tohoto identifikátoru je v rozmezí 5-16 bajtů. Prvních 5 bajtů je pole RID (registrovaný aplikační identifikátor) a za ním může následovat 0-11 bajtů pole PIX (proprietární rozšíření identifikátoru). Protože je možné, aby jeden druh appletu měl na kartě několik svých instancí, jsou každému appletu ve skutečnosti přiřazeny AID dvě. Jedno udává AID balíčku, druhé udává AID samotného appletu [6]. Tvorba těchto AID probíhá vždy při překladu.

2.2.4 Instalace appletu na kartu

Před nahráním appletu na kartu je nutné tento applet přeložit do formátu, kterému bude čipová karta rozumět. Ze zdrojového kódu v programovacím jazyce Java je nejdříve Java kompilátorem vytvořen .class soubor, tento soubor je poté společně s přiřazeným AID převeden na soubor .cap pomocí Java Card konvertoru, který je součástí JCAPI. Výsledný soubor `.cap` je pak již možné nahrát na kartu.

Pro nahrání na kartu je možné využít různých metod. Některá vývojová prostředí umí nahrávat applety na kartu přímo (např. **JCSuite 3.0** který je k dipozici ke kartám Sm@rtCafé od firmy Giesecke&Devrient). Tento proces je velice intuitivní a snadný. Další možností je použít aplikaci **GPShell**, která disponuje velkým počtem různých funkcí pro komunikaci s kartou. Nevýhodou GPShell je ale nutnost psát

veškeré příkazy ručně a uživatel musí mít dobré znalosti komunikace s kartou na úrovni APDU příkazů. Příkazy je také možné definovat do skriptů a tyto skripty pak hromadně vykonávat. Další možností je využít aplikace **GlobalPlatformPro** [13] (dále jen GP), která je jakousi uživatelsky přívětivější formou GPShell. Nedisponuje sice takovou škálou funkcí, ale je snažší na používání a pro obyčejnou instalaci appletu na kartu nebo smazání appletu z karty je nejjednodušší a nejrychlejší. Proto jsem tuto aplikaci při své práci využíval nejvíce.

Nejvíce využívanými příkazy programu GP jsou `gp -info`, které zobrazí informace o kartě, jako například parametry, verzi JCAPI apod. Dalším velice užitečným příkazem je `gp -list`, který vypíše instalované applety na kartě, včetně jejich AID. Samotná instalace na kartu pomocí GP je velice snadná. Lze ji spustit z příkazové řádky příkazem `gp -install <název_souboru>` – tedy nainstaluj tento applet na kartu. Při instalaci proběhne kontrola, zda není příslušné AID již na kartě přítomno a pokud není, nahraje soubor na kartu a vykoná příslušné operace pro uložení appletu v kartě a zaznamenání jeho AID v JCRE. Od této chvíle je pak možné pomocí APDU příkazů select volit applety na kartě a následně s nimi komunikovat pomocí APDU příkazů. Pro smazání appletu z karty slouží příkaz `gp -uninstall <název_souboru>`, který ze souboru zjistí AID appletu, smaže příslušný applet z karty (pokud se tam nachází) a odregistruje jej z JCRE.

2.2.5 Překlad appletů požadovaným JavaCard API

Z důvodu toho, že různé karty podporují různé verze JCAPI, je důležité applety překládat pomocí adekvátní verze API. Většina vývojových prostředí pracuje vždy s nejnovější verzí API a pokud potřebuje uživatel provést překlad nějakou ze starších verzí, bývá toto nastavení dosti matoucí a obtížné. Protože karty které jsem měl při této práci k dispozici pracuji s API verze 2.2.1, musel jsem tento problém překladu také nějak vyřešit.

Nejlepším způsobem jak dosáhnout překladu ve správné verzi API bylo použití jednoduché knihovny **ant-javacard** [12], kterou stačí ve vývojovém prostředí nalinkovat k tvořenému appletu a pak už jen v souboru build.xml zadat cestu ke zdrojovým souborům požadované verze JCAPI, pomocí které chceme applet přeložit. Není potřeba nic složitě nastavovat ve vývojovém prostředí a v případě potřeby překladu jinou verzí API stačí pouze drobná změna cesty v build.xml a vše je připraveno k novému překladu pomocí jiné verze.

3 KRYPTOGRAFIE NA JAVA CARD

Platforma Java Card se postupným vývojem neustále obohacuje o nové kryptografické prostředky a nabízí tak podporu pro vývoj nových a bezpečnějších aplikací. První specifikace JCAPI byla vydána v roce 1999 a jednalo se o verzi 2.1, která obsahovala pouze operace s šiframi DES, DSA, také RSA pro klíče o maximální délce klíče 1024b a pár podepisovacích algoritmů postavených na zmíněných šifrách.

Další významnou verzí byla specifikace 2.2, která přinášela mimo jiné operace s AES, CRC, ale hlavně přibyla podpora eliptických křivek a protokolu Diffie-Hellman. Také byla přidána podpora delších klíčů pro RSA. Následné dodatky k této verzi, označované jako 2.2.1 a 2.2.2 doplňovaly hlavně podporu dalších typů paddingu pro jednotlivé algoritmy a také rozšíření hašovacích funkcí. Tyto dvě podverze jsou aktuálně asi nejvíce rozšířené a čipové karty podporující tyto specifikace jsou dobře dostupné taktéž cenově - avšak vše záleží na daném výrobci a na množství podporovaných algoritmů u daného modelu karty.

V roce 2008 pak byla vydána nová verze API s označením 3.0, které je již rozdělená na dvě části - Classic Edition a Connected Edition. Zatímco Classic Edition dále rozšiřuje předchozí specifikace o další nové algoritmy, délky klíčů a jiné, Connected Edition představuje nový virtuální stroj JCVM pro podporu nové komunikace nahrazující předchozí APDU, kde se bude využívat protokolu postaveném na HTTP[1].

3.1 Dostupné kryptografické prostředky karet

Zjišťování kryptografických prostředků, kterými java karty disponují může být občas složité. Ne vždy je ke kartě dostupný datasheet se všemi podporovanými a nepodporovanými algoritmy a prostředky. V takových případech je pak dost obtížné vyvíjet na této platformě applety, kvůli absenci snadného debugování. Při hledání způsobu, jak jednoduše a rychle určit kryptografické možnosti java karet jsem na internetu narazil na velice užitečný nástroj Petra Švendy - JavaCard algorithms support test[14].

Jedná se kolekci appletů zkonstruovaných tak, aby ověřili dostupnost všech možných algoritmů dostupných ve specifikaci JCAPI. Tento nástroj stačí pouze nahrát na kartu a pomocí přiložené konzolové aplikace spustit analýzu karty. Protože existuje několik různých verzí JCAPI a každá karta podporuje pouze určitou verzi, je zde pro každou verzi JCAPI nachystaný příslušný applet, takže odpadá problém s nekompatibilitou appletu a JCAPI na kartě. Po dokončení testu je k dispozici vygenerovaný .csv soubor s kompletním souhrnem dostupných algoritmů na kartě.

[1] Všechny změny které byly v jednotlivých verzích JCAPI prováděny, včetně detailních popisů jednotlivých tříd a metod je možné dohledat ve specifikacích na adrese: `http://www.oracle.com/technetwork/java/embedded/javacard/downloads/index.html`

Samotný test trvá přibližně kolem minuty, v závislosti na rychlosti karty a na počtu dostupných algoritmů.

Tento nástroj velice usnadní práci s kartami a dává k dispozici výborný přehled dostupných algoritmů na kartách, se kterými se dá pracovat. V tabulkách A.1,A.2,A.3 a A.4 je souhrn testovaných karet a jimi podporovaných algoritmů. Pro přehlednost a stručnost jsou uvedeny pouze dostupné algoritmy. Z tohoto souhrnu jsem mohl pak pohodlně určit, s jakými kryptografickými algoritmy budu při návrhu autentizačního schématu pracovat.

3.2 Kryptografie nad eliptickými křivkami

V kryptografii nad eliptickými křivkami se využívá algebraických struktur eliptických křivek nad konečnými tělesy. Konečná tělesa se vyznačují tím, že počet prvků v nich je konečný a lépe se tak hodí pro aplikaci v kryptografii, oproti eliptickým křivkám definovanými nad reálnými čísly. Eliptická křivka je definována rovnicí:

$$E : y^2 + 2xy = ax^3 + bx^2 + cx + d \tag{3.1}$$

Častěji se však setkáme s upravenou rovnicí ve Weierstrassově tvaru:

$$E : y^2 = x^3 + ax + b \tag{3.2}$$

Hodnoty a a b jsou koeficienty rovnice a aby byla eliptická křivka nesingulární (tzn. se jednalo o eliptickou křivku), musí platit následující hlavní podmínka:

$$4a^3 + 27b^2 \neq 0 \tag{3.3}$$

Hodnoty x a y potom udávají souřadnice všech bodů ležících na dané křivce, které vyhovují rovnici 3.2. V kryptografii se nejčastěji setkáme s eliptickými křivkami definovanými nad konečným tělesem F_p, kde p je velké prvočíslo definující těleso F_p a také se můžeme setkat se speciální variantou F_{2m}, což naznačuje, že se jedná o binární řetězce ($p = 2$) o délce m. U této varianty eliptických křivek se volí m jako prvočíslo a místo prvočísla p se pak v parametrech křivky udává funkce $f(x)$ specifikující polynomiální bázi tělesa o stupni m [5].

Každá křivka má definovaný svůj řád, značený $\#E$, který udává celkový počet bodů této křivky, dále bod $\mathcal{O}$ neboli bod v nekonečnu a také generující bod G (také označovaný jako P). Na těchto dvou bodech je pak závislý další parametr nazvaný řád bodu G značený jako n, pro který platí $n \cdot G = \mathcal{O}$ a také je dělitelem řádu křivky $\#E$. Posledním, v kryptografii udávaným, parametrem je ještě kofaktor h

definovaný jako $h = \frac{\#E}{n}$, který se však používá pouze u křivek F_{2m} (u křivek F_p je $h = 1$, protože $\#E = n$). Pro křivky typu F_{2m} pak platí ještě jedna dodatečná podmínka pro rovnici č 3.3, a to že $b \neq 0$.

Eliptická křivka E je tedy v kryptografii definována:

- Pro F_p jako šestice $E_{F_p} = (p, a, b, G, n, h)$
- Pro F_{2m} jako sedmice $E_{F_{2m}} = (m, f(x), a, b, G, n, h)$

Protože se jedná o asymetrický kryptosystém, figuruje v něm pochopitelně soukromý a veřejný klíč. Soukromý klíč d je náhodně vygenerované číslo v intervalu $1 < d < (n-1)$ a veřejný klíč Q je potom vypočítaný jako $Q = d \cdot G$.

3.2.1 ECC na platformě Java Card

Platforma Java Card je na použití eliptických křivek náležitě připravená a disponuje podporou křivek nad tělesy F_p i F_{2m}. Specifikace JCAPI 2.x měla k dispozici celkem 8 druhů křivek – pro F_{2m} křivky $m = 113, 131, 163, 193$ a pro F_p křivky $p = 112, 128, 160, 192$. Od verze JCAPI 3.x pak dále přibyly další 4 druhy křivek pro F_p, konkrétně $p = 224, 256, 384, 521$. Je také důležité zmínit, že velmi záleží na konkrétním výrobci a druhu čipové karty – některé karty eliptické křivky nemusí podporovat vůbec nebo podporují pouze omezený výběr křivek.

Implementace a práce se samotnými křivkami na kartách je pak možné rozdělit na dva druhy přístupu. Buď je možné využít už připravených parametrů křivek, kterými daná čipová karta disponuje, nebo je možné si definovat křivku vlastní. V případě, že čipová karta disponuje již připravenými parametry křivek, je nejjednodušší a nejsnažší využít třídy `KeyPair` pro vygenerování soukromého a veřejného klíče. Při inicializaci pouze specifikujeme dva parametry - jakou křivku chceme použít, jestli F_p nebo F_{2m} a dále pak velikost křivky (klíče). Po zavolání metody `genKeyPair()` pak již dostáváme veřejný a soukromý klíč zvolené křivky, které jsou při každém volání jiné, čímž se vyvarujeme ukládání jejich hodnot v plaintextové formě v kódu appletu. Takto vygenerované klíče pak můžeme použít pro ECDH, ECDSA a další algoritmy.

V případě, že chceme využít vlastních parametrů křivky, je postup dosti odlišný. Jako první je potřeba si vytvořit zvlášť objekty pro soukromý (`ECPRivateKey`) a veřejný (`ECPublicKey`) klíč a těmto klíčům potom pomocí připravených set-metod nastavit příslušné parametry námi požadované křivky. Tedy p (případně m a $f(x)$ u typu F_{2m}), a, b, G, n, a h. Když jsou tyto parametry u obou objektů nastavené, je potřeba také nastavit hodnotu soukromého a veřejného klíče. Zde je oproti předchozímu případu jasné, že tyto hodnoty soukromého a veřejného klíče je zapotřebí mít na kartě buď napevno uložené (jsou předsdílené a nemění se) a nebo je potřeba tyto hodnoty na kartu nějakým způsobem odeslat, čímž se ale vystavujeme riziku úniku

zvláště soukromého klíče útočníkovi. Když jsou i tyto hodnoty úspěšně nastaveny, jsou již klíče připravené k použití v požadovaných algoritmech.

3.2.2 Parametry křivek a jejich typy na platformě Java Card

Při vývoji appletu pro čipové karty jsem se snažil pracovat hlavně s třídou `KeyPair` pro generování klíčů přímo na kartě, díky které jsou vygenerované klíče vždy jiné. Aby však bylo možné s těmito klíči dále pracovat v ECDH protokolu v komunikaci s terminálem, bylo důležité zjistit, jaké předpřipravené parametry křivek se vlastně na kartách nacházejí. Kromě metod na nastavení parametrů křivek obsahuje JCAPI také metody na získání parametrů křivek. Postupným testováním jednotlivých druhů křivek a odesíláním jejich parametrů na terminál jsem ve výsledku zjistil, že všechny karty NXP JCOP, které jsem měl v době tvorby appletu k dispozici a zároveň podporují eliptické křivky, disponují křivkami specifikovanými v dokumentu SEC2 [7]. Konkrétně se jednalo o karty:

- NXP JCOP 31 v2.2 72K (pouze křivky F_{2m}, křivky F_p nepodporuje vůbec)
- NXP JCOP 31 V2.3.2 32K (pouze křivky F_{2m}, křivky F_p nepodporuje vůbec)
- NXP JCOP J3A040 CL (pouze křivky F_p, křivky F_{2m} nepodporuje vůbec)

Přesněji řečeno jsem testováním parametrů generovaných klíčů pomocí `KeyPair` došel k následujícím závěrům.

Tabulka 3.1 obsahuje jména jednotlivých křivek, podle toho, jak jsou pojmenovány ve výše zmíněném dokumentu, kde je poté snadné si dohledat veškeré parametry daných křivek. Výstupem posledního kroku algoritmu Diffie-Hellman je pochopitelně sdílený klíč v délce určené použitou eliptickou křivkou. Na platformě Java Card se tento výsledný sdílený klíč z bezpečnostních důvodů negeneruje v plaintextové podobě, ale je navíc zhašovaný algoritmem SHA-1. Tento krok není volitelný a je prováděn automaticky v posledním kroku. To ve výsledku znamená, že sdílené klíče jsou vždy v délce 20 bajtů, od čehož se také odvíjí nutnost provádět stejné hašování sdíleného klíče na straně terminálu.

Křivky nad tělesy F_{2m}		Křivky nad tělesy F_p	
Velikost křivky	Název dle SEC2	Velikost křivky	Název dle SEC2
F_{2m} 113	sect113r1	F_p 112	secp112r1
F_{2m} 131	sect131r1	F_p 128	secp128r1
F_{2m} 163	sect163r1	F_p 160	secp160r1
F_{2m} 193	sect193r1	F_p 192	secp192r1

Tab. 3.1: Tabulka názvů křivek dle SEC2v1 na kartách NXP JCOP

3.2.3 Rychlost operací nad eliptickými křivkami na čipových kartách

Rychlosti výpočtů nad eliptickými křivkami je samozřejmě dána hardwarovými dispozicemi používané čipové karty – pracovní frekvence karty, rychlost koprocesoru, velikost paměti. Časy se taktéž lišily v závislosti na tom, zda je využito generování klíčů přímo na kartě pomocí třídy `KeyPair` nebo zda jsou použity vlastní parametry křivky. Tabulka 3.2 ukazuje průměrné hodnoty jednotlivých operací u 4 různých testovaných karet.

NXP JCOP 31 v2.2 72K				
Typ eliptické křivky	KeyPair generace klíčů	Načtení vlastních parametrů EC	Inicializace ECDH soukromým klíčem	ECDH generace sdíleného klíče
F_{2m} 113	703 ms	660 ms	138 ms	78 ms
F_{2m} 131	710 ms	655 ms	137 ms	81 ms
F_{2m} 163	—	668 ms	137 ms	90 ms
F_{2m} 193	—	675 ms	141 ms	103 ms

NXP JCOP 31 v2.3.2 32K				
Typ eliptické křivky	KeyPair generace klíčů	Načtení vlastních parametrů EC	Inicializace ECDH soukromým klíčem	ECDH generace sdíleného klíče
F_{2m} 113	719 ms	656 ms	145 ms	78 ms
F_{2m} 131	738 ms	660 ms	132 ms	84 ms
F_{2m} 163	—	662 ms	151 ms	97 ms
F_{2m} 193	—	678 ms	146 ms	100 ms

NXP JCOP J3A040 CL				
Typ eliptické křivky	KeyPair generace klíčů	Načtení vlastních parametrů EC	Inicializace ECDH soukromým klíčem	ECDH generace sdíleného klíče
F_p 112	—	—	—	—
F_p 128	736 ms	515 ms	110 ms	97 ms
F_p 160	764 ms	521 ms	104 ms	106 ms
F_p 192	824 ms	529 ms	109 ms	113 ms
F_p 224	—	532 ms	109 ms	137 ms
F_p 256	—	534 ms	106 ms	190 ms

SmartCafe Expert 6.0				
Typ eliptické křivky	KeyPair generace klíčů	Načtení vlastních parametrů EC	Inicializace ECDH soukromým klíčem	ECDH generace sdíleného klíče
F_p 112	—	194 ms	28 ms	75 ms
F_p 128	—	200 ms	25 ms	75 ms
F_p 160	—	203 ms	25 ms	97 ms
F_p 192	—	222 ms	31 ms	122 ms
F_p 224	—	207 ms	32 ms	153 ms
F_p 256	—	206 ms	32 ms	188 ms

Tab. 3.2: Tabulka průměrných časů operací nad eliptickými křivkami

Byly měřeny časy generace klíčů přímo na kartě pomocí předpřipravených křivek SEC2, tedy generace soukromého i veřejného klíče pro ECDH algoritmus. Jak bylo zmíněno, karta SmartCafe Expert 6 od společnosti Gieseck&Devrient generovat klíče neumí, protože nedisponuje žádnými předpřipravenými křivkami.

Druhým měřeným časem bylo nastavení všech potřebných hodnot pro použití vlastních parametrů křivky - parametry křivky a také načtení soukromého a veřejného klíče z bajtového pole.

Třetí měřenou hodnotou byl čas potřebný pro inicializace ECDH algoritmu soukromým klíčem u třídy `KeyAgreement`. Poslední měřenou hodnotou bylo generování sdíleného klíče taktéž třídou `KeyAgreement`.

Zobrazené časy jsou výsledkem aritmetických průměrů pětice měření, políčka bez údajů naznačují, že daný druh křivky nebo danou funkci karta nepodporuje. Z naměřených výsledků je možné vidět, že poměrně nová karta SmartCafe Expert 6 je svým výkonem téměř dvakrát rychlejší než karty NXP JCOP.

4 AUTENTIZAČNÍ PROTOKOL PACE

Password Authenticated Connection Establishment, dále už jen ve zkratce jako PACE, je heslem ověřený protokol pro bezpečné ustanovení šifrovacích klíčů pro následnou zabezpečenou komunikaci. Základním stavebním kamenem tohoto protokolu je fakt, že celý protokol se skládá z několika na sebe navazujících dílčích operací a kombinují tak několik různých kryptografických postupů pro ustanovení výsledných klíčů na základě znalosti jen dvou počátečních hodnot. V PACE protokolu je obsažena klasická symetrická kryptografie pro šifrovaní a dešifrování jednoduchých zpráv, dále je zde využito kryptografie eliptických křivek v kombinaci s Diffie-Hellman alogritmem, pak také hašovací funkce a v neposlední řadě ještě digitální podpis MAC. PACE protokol se v současnosti používá například v čipových občanských průkazech v Německu a jeho specifikaci popsal taktéž německý spolkový úřad pro inofrmační bezpečnost [8]. Následující podkapitoly popisují v souhrnu samotný protokol PACE a dále pak v detailech jeho dílčí části se zaměřením na to, jak jsou tyto části aplikovaný do prostředí Java Card.

Součástí protokolu PACE je část vzájemného ustanovení základních klíčů, kterou je možné implementovat několika různými způsoby. Jedná se o část Map2Point [9]. Tato práce se nevěnuje všem možným variantám, ale pouze variantě DH2Point, která využívá Diffie-Hellmanova algoritmu nad eliptickými křivkami.

4.1 Dílčí části PACE protokolu

Nejdříve je zapotřebí stanovit, jaké předsdílené parametry jsou oběma protistranám známy před samotným započetím celého algoritmu. Tyto parametry jsou v širší podstatě pouze dva:

1. Předsdílené počáteční heslo π
2. Parametry eliptické křivky $\mathcal{G}$

Přesněji řečeno je však parametrů celkem 7, protože eliptické křivka je definována jako šestice, $\mathcal{G} = (p, a, b, G, n, h)$. Tyto předsdílené parametry mají obě strany uložené v paměti a na jejich základě protokol PACE staví. Předsdílené heslo π je použito pouze v prvním kroku PACE protokolu, s eliptickou křivkou je pak dále pracováno v kroku druhém. Pokud dojde v některé z částí PACE protokolu k chybě, je jeho vykonávání přerušeno a doposud získané hodnoty jsou již neplatné.

Obrázek 4.1 graficky znázorňuje průběh PACE protokolu, který lze rozdělit na 5 dílčích částí, v postupném pořadí jdoucím po sobě, následovně:

1. Vygenerování náhodné hodnoty na straně PC, zašifrování a její odeslání a uložení na čipovou kartu
2. Provedení ECDH algoritmu nad definovanou eliptickou křivkou
3. Provedení upravené varianty DH algoritmu
4. Tvorba finálních klíčů hašováním
5. Ověření digitálních podpisů obou protistran a potvrzení platnosti klíčů

A		B
password π		password π
authenticated EC parameters $\mathcal{G} = (a, b, p, q, G, k)$		
$K_\pi = \mathcal{H}(\pi\|\|0)$		$K_\pi = \mathcal{H}(\pi\|\|0)$
choose $s \leftarrow \mathbb{Z}_q$		
$z = \mathcal{C}(K_\pi, s)$		
	$\xrightarrow{\mathcal{G}, z}$	abort if $\mathcal{G}$ incorrect
		$s = \mathcal{C}^{-1}(K_\pi, z)$
.....	$\mathsf{Map2Point}(s)$	.....
	$\rightleftharpoons$	
	jointly generate $\widehat{G}$	
choose $y_A \leftarrow \mathbb{Z}_q^*$		choose $y_B \leftarrow \mathbb{Z}_q^*$
$Y_A = y_A \cdot \widehat{G}$		$Y_B = y_B \cdot \widehat{G}$
	$\xleftarrow{Y_B}$	
abort if $Y_B \notin \langle G \rangle \setminus \{0\}$		
	$\xrightarrow{Y_A}$	abort if $Y_A \notin \langle G \rangle \setminus \{0\}$
$K = y_A \cdot Y_B$		$K = y_B \cdot Y_A$
$K_{\text{enc}} = \mathcal{H}(K\|\|1)$		$K_{\text{enc}} = \mathcal{H}(K\|\|1)$
$K_{\text{mac}} = \mathcal{H}(K\|\|2)$		$K_{\text{mac}} = \mathcal{H}(K\|\|2)$
$K'_{\text{mac}} = \mathcal{H}(K\|\|3)$		$K'_{\text{mac}} = \mathcal{H}(K\|\|3)$
$T_A \leftarrow \mathcal{M}(K'_{\text{mac}}, (Y_B, \mathcal{G}))$		$T_B \leftarrow \mathcal{M}(K'_{\text{mac}}, (Y_A, \mathcal{G}))$
	$\xleftarrow{T_B}$	
	$\xrightarrow{T_A}$	
abort if T_B invalid		abort if T_A invalid
key= $(K_{\text{enc}}, K_{\text{mac}})$		key= $(K_{\text{enc}}, K_{\text{mac}})$
sid = $(Y_A, Y_B, \mathcal{G})$		sid = $(Y_A, Y_B, \mathcal{G})$
pid = ϵ		pid = ϵ

Obr. 4.1: Grafické znázornění průběhu PACE protokolu[9]

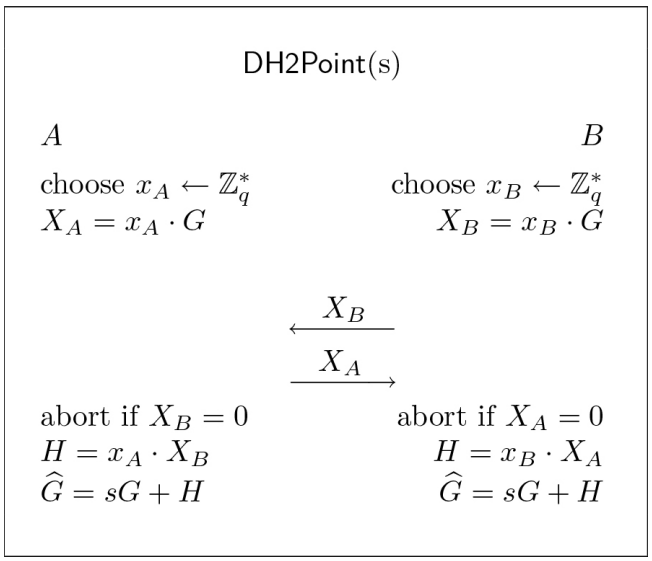
4.2 První část protokolu PACE

Jako první obě protistrany musí vytvořit haš předsdíleného hesla, aby získaly klíč K_π. Jako předsdílené heslo používám 16bajtové pole. Toto pole je před samotným hašováním rozšířeno na 17bajtové pole doplněním nuly na konec původního pole. Následně je toto pole zhašováno algoritmem **SHA**, čímž dostávám 20bajtové pole s hašem. Z tohoto haše je pak prvních 16bajtů použito jako klíč pro symetrickou šifru **AES**, poslední 4 bajty jsou zahozeny. Tedy obě protistrany vytvoří $K_\pi = h_{SHA}(\pi \| 0)$.

Následně na straně PC dochází k vygenerování, zašifrování a odeslání náhodné hodnoty s na kartu, kde je opět dešifrována a uložena. Protože tato náhodná veličina figuruje později v PACE protokolu v násobení, má velikost této hodnoty zásadní vliv na rychlost celého protokolu. V této práci generuji hodnotu s jako 16bajtové náhodné pole. Tuto hodnotu potom pomocí klíče K_π zašifruje a odešle na kartu kryptogram z, tj. $z = E_{AES}(s, K_\pi)$. Karta obdrží z a stejným klíčem K_π si zprávu dešifruje a uloží si vygenerovanou hodnotu s. Přitom také ověří, zda $s \neq 0$. Pokud by bylo $s = 0$, je provádění PACE protokolu ukončeno.

4.3 Druhá část protokolu PACE

Poté, co obě protistrany znají náhodnou hodnotu s, provedou funkci **Map2Point**, v mém případě variantu **DH2Point**, využívající algoritmu Diffie-Hellman nad eliptickou křivkou (dále jen ECDH). Princip algoritmu je znázorněn na 4.2. Postup je takový, že obě protistrany si zvolí svůj soukromý klíč, v případě ECDH se tedy jedná o nějaký bod na křivce $\mathcal{G}$. Tedy $x_A \in \mathcal{G}$ resp. $x_B \in \mathcal{G}$.



Obr. 4.2: Grafické znázornění algoritmu DH2Point[9]

Pomocí těchto klíčů (bodů) a generátoru křivky G pak následně vytvoří veřejné klíče jako $X_A = x_A \cdot G$ resp. $X_B = x_B \cdot G$. Tyto veřejné klíče (opět body na křivce) si protistrany vymění a přitom také zkontrolují zda od sebe vzájemně neobdržely nulovou hodnotu – zda $X_A \neq 0$ nebo $X_B \neq 0$. Pokud ano, je provádění PACE protokolu ukončeno. Pokud jsou hodnoty nenulové, pomocí násobení dvou bodů na křivce – vlastního soukromého klíče a veřejného klíče protistrany vytvoří finální sdílený ECDH klíč, který pojmenujme jako H, přičemž $H = x_A \cdot X_B = x_B \cdot X_A$.

Získané H se ještě poté přičte k výsledku násobení mezi hodnotou generátoru G a náhodnou hodnotou s, čímž obě protistrany získají $\hat{G}$. Zapsáno rovnicí jako $\hat{G} = s \cdot G + H$. V tomto případě už se jedná o klasické aritmetické sčítání a násobení.

4.4 Třetí část protokolu PACE

V této části figuruje nemodulární varianta klasického Diffie-Hellman protokolu. Jako generátor je použito $\hat{G}$ z předchozího kroku. Obě protistrany si opět vyberou nějaké soukromé náhodné číslo y_A resp. y_B a vypočtou klasickým aritmetickým násobením veřejná čísla $Y_A = y_A \cdot \hat{G}$ a $Y_B = y_B \cdot \hat{G}$. Opět dojde k výměně těchto veřejných čísel, jejich kontrole zda nejsou nulová (pokud ano, PACE protokol končí) a následnému výpočtu klíče K jako $K = y_A \cdot Y_B = y_B \cdot Y_A$.

4.5 Čtvrtá část protokolu PACE

Zde již dochází ke generování finálních klíčů PACE protokolu. Získané K z předchozího kroku je zde použito pro vytvoření tří klíčů K_{ENC}, K_{MAC} a K'_{MAC}. Tyto klíče jsou tvořený tak že je pro každý jeden z klíčů doplněn klíč K o číslo 1, 2 resp. 3 a následně zhašován pomocí funkce SHA.

- $K_{ENC} = h_{SHA}(K \| 1)$
- $K_{MAC} = h_{SHA}(K \| 2)$
- $K'_{MAC} = h_{SHA}(K \| 3)$

Klíče K_{ENC}, K_{MAC} jsou na konci provádění PACE protokolu použity jako výsledné dohodnuté klíče pro následnou šifrovanou komunikaci. Aby bylo možné prohlásit PACE protokol jako úspěšně dokončený, je však zapotřebí dokončit poslední krok.

4.6 Pátá část protokolu PACE

V poslední části PACE protokolu je u obou protistran pomocí klíče K'_{MAC}, parametrů křivky $\mathcal{G}$ a veřejného klíče Y_A resp. Y_B (z části číslo 3) vytvořen MAC podpis T, který si protistrany vymění a ověří jejich platnost. Vyjádřeno vzorci,

$T_A = MAC((\mathcal{G}, Y_B), K'_{MAC})$ a $T_B = MAC((\mathcal{G}, Y_A), K'_{MAC})$. Pokud některá z protistran obdrží neplatný podpis, je protokol PACE ukončen. Pokud jsou podpisy obou stran správné, protokol PACE končí a klíče K_{ENC} a K_{MAC} se považují za finální klíče pro následnou šifrovanou komunikaci mezi protistranami. PACE protokol také stanovuje sessionID jako trojici $sid = (Y_A, Y_B, \mathcal{G})$, které je možné při následné komunikaci také kontrolovat, ale v rámci této práce se sid nepracuji a používám pouze výsledný K_{ENC} jako nový klíč pro symetrickou šifru AES pro budoucí komunikaci mezi protistranami.

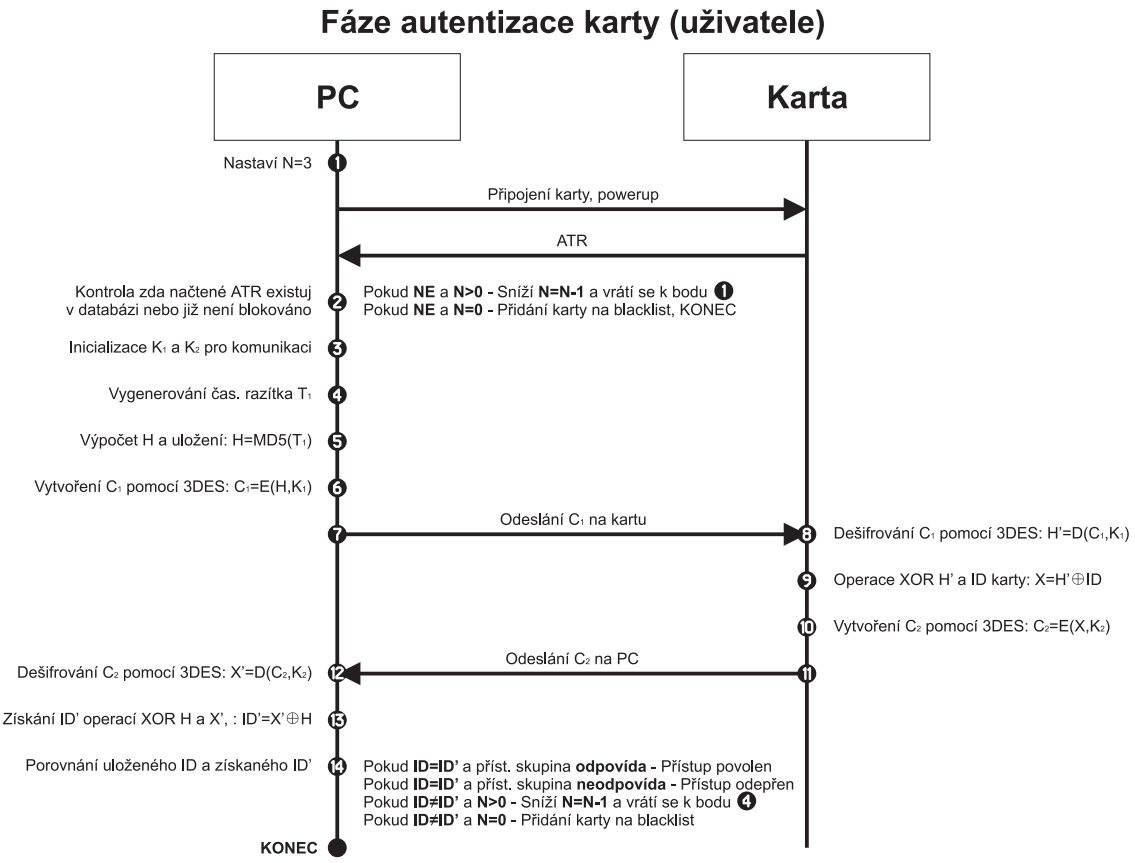
5 REALIZACE PŘÍSTUPOVÉHO SYSTÉMU

Kritériem kladeným při návrhu přístupového systému byla jeho bezpečnost. Bylo snahou co nejvíce využít potenciál čipových karet a zároveň pracovat s pokročilými kryptografickými schématy. Při analýze dostupných platforem pro návrh systému jsem pracoval s produkty několika výrobců, jako například JavaCard nebo MULTOS, u kterých lze uplatnit programování v jazyce Java nebo třeba karty .NET SmartCard pracující s programovacím jazykem C# (Microsoft .NET). Nakonec jsem zvolil pouze platformu Java Card. Hlavním důvodem byla kvalitní podpora eliptických křivek, která u MULTOS karet chyběla. Navíc tím, že je vše psáno v programovacím jazyce Java, bylo by snadné systém rozšířit o podporu mobilních aplikací pro telefony s technologií NFC, kde je možné telefon pomocí emulací použít uplně stejně jako čipovou kartu. Na konci této kapitoly se zaměřím na porovnání předchozího a nového návrhu se zaměřením hlavně na bezpečnost a také na rychlost celého systému.

5.1 Původní návrh přístupového systému

Návrh systému realizovaného v rámci semestrální práce stavěl celou autentizaci na znalosti jednoho hlavního klíče systému a jednotlivých přidělených klíčů karet, také využíval pro každou autentizaci časového razítka, hašovací funkci MD5 a dále pro ověřování identity a určení správného klíče karty používal ATR záznamů samotných čipových karet. Systém, který autentizaci prováděl, měl lokálně uložen hlavní klíč, používaný pro šifrování zpráv zasílaných z počítače na kartu. Dále byů pro každého uživatele (kartu) veden záznam v databázi nesoucí ID uživatele, ATR přidělené karty a také unikátní 24B klíč této kartě přidělený pro šifrování zpráv zasílaných z karty do počítače.

Databáze uživatelů byla řešena pouze lokálně uloženým XML souborem nesoucím údaje o uživatelích. Samotné ověřování uživatele systémem je znázorněno na obrázku č. 5.1. Systém dával prostor pro 3 opakování autentizace. Pokud ani na třetí pokus nebyla autentizace úspěšná, byla daná karta (uživatel) podle jejího ATR v systému zablokována a tím je zamezena jakékoliv další komunikaci s kartou.



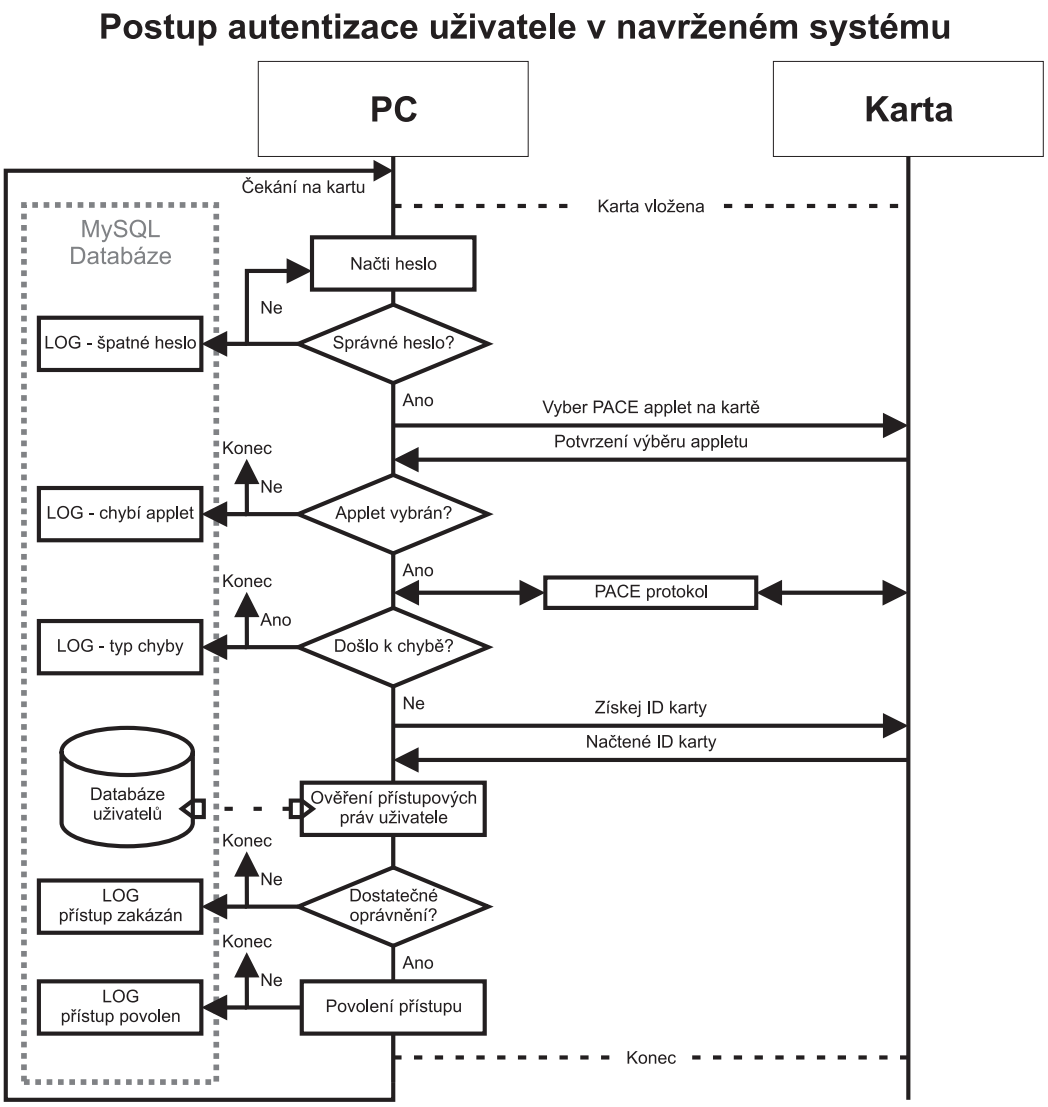
Obr. 5.1: Postup autentizace karty systémem podle původního návrhu

5.2 Přístupový systém využívající PACE protokol

Nově navržený systém se skládá z několika prvků a na sebe navazujících kroků. Celý autentizační proces je zahájen dotazem na počáteční heslo terminálovou aplikací. Dále následuje provedení PACE protokolu, po jehož úspěšném dokončení je možné se čipové karty dotázat na ID uživatele, které je na kartě uloženo. Následuje načtení informací o uživateli ze vzdálené databáze a nakonec porovnání přístupových práv a udělení/zamítnutí přístupu. Obrázek č. 5.2 graficky naznačuje průběh autentizačního procesu v navrženém systému. Při zjišťování, zda má uživatel dostatečná práva k přístupu se porovnává číselný údaj přístupové skupiny (dále jen GID). Tento údaj si může každý přístupový terminál zvolit vlastní (kladná nenulová hodnota), přičemž platí, že čím menší číslo GID, tím vyšší přístupová práva uživatel má. Při porovnávání se terminál řídí následovně:

- Pokud $GID_{terminálu} < GID_{uživatele}$, přístup **není povolen**
- Pokud $GID_{terminálu} \geq GID_{uživatele}$, přístup **je povolen**

Z toho plyne, že uživatel s $GID = 1$ má přístup všude a uživatel s $GID = 3$ má přístup pouze k terminálům s $GID \geq 3$.



Obr. 5.2: Autentizace uživatele v navrženém přístupovém systému

Následující podkapitoly se budou podrobně věnovat jednotlivým částem systému. Rozbor funkčnosti a prvků navrženého appletu pro platformu Java Card, popis struktury a funkčnosti terminálové aplikace s uvedením rozdílů mezi správcovskou a uživatelskou verzí aplikace a také popis navržených databází a aplikace pro zobrazování dat z databáze.

Ochrana terminálu heslem

Jak je patrné, nově navržený přístupový systém staví převážné na samotném protokolu PACE. Kromě toho je systém opatřen o několik dalších doplňků, které ještě zvyšují jeho zabezpečení. Před začátkem autentizace uživatele je systém doplněn o dotaz na počáteční heslo. Toto heslo chrání systém i uživatele proti ztrátě nebo odcizení čipové karty. Znalost hesla má každý uživatel ve své paměti a bez tohoto

hesla není možné ani započít proces samotné autentizace. Výhodou tohoto řešení je také fakt, že v případě požadavků na vyšší bezpečnost je možné toto heslo libovolně často měnit a provozovatel systému také může libovolně určit jeho sémantickou složitost - od jednoduchého číselníku, tedy hesla skládajícího se pouze z numerických znaků, až po složité alfanumerické řetězce s přídavkem symbolů a znaků mimo standardní abecedu.

Ochrana appletu na čipové kartě

Dále je také aplikována ochrana v rámci appletu na čipové kartě. Applet je opatřen počítadlem špatných autentizačních pokusů. V případě obdržení špatné délky očekávaných dat, obdržení nulových dat, špatného MAC podpisu nebo například obdržení špatného typu APDU příkazu (nepodporované CLA, INS, vyžádání ID bez předchozího úspěšného provedení PACE) dochází uvnitř karty k dekrementaci počítadla špatných pokusů. Toto počítadlo si vydavatel karet můžu při překladu appletu zvolit taktéž libovolně (aktuálně nastaveno na 5 pokusů). Pokud dojde k vyčerpání špatných pokusů, dojde k automatickému zablokování appletu na kartě a pro opětnou aktivaci je potřeba applet z karty odstranit a znovu nainstalovat u důvěryhodného vydavatele těchto karet (například správce přístupového systému). Při návrhu této ochrany jsem také dbal na různé možné varianty útoku, jako například vytvoření nové instance appletu, které by mohlo mít za důsledek resetování tohoto počítadla a umožnění dalších pokusů o útok na applet. Tuto variantu jsem vyřešil deklarováním počítadla datovým typem static, který zajišťuje, že nové instance appletu budou uchovávat původní hodnotu počítadla – tedy pokud je applet již zablokovaný, tak i jeho nová instance appletu bude zablokována. Pokud celý autentizační proces proběhne úspěšně, je počítadlo resetováno do původního stavu pokusů, aby se zamezilo omylnému zablokování karty například v případě chybné komunikace mezi čtečkou a kartou nebo například chybám na straně terminálové aplikace.

Ochrana databáze uživatelů

Dalším důležitým vylepšením oproti původnímu návrhu je databáze uživatelů. Uchovávat databázi uživatelů lokálně, navíc bez jakéhokoliv šifrování, vystavuje systém jak možnosti útoku přímo na soubory databáze. Timto se systém stává velice nemodulárním a v případě několika desítek terminálů rozprostřených po objektu také složitým na údržbu a aktuálnost samotných databází s uživateli. Velmi vhodným řešením je mít celou databázi centralizovanou v jednom místě a přistupovat k ní vzdáleně. Proto jsem v novém návrhu šál po SQL databázi, který podporuje za-

bezpečenou komunikaci. Zvolil jsem konkrétně MySQL server [1], který podporuje SSL/TLS komunikaci, ale aktuální trh nabízí nespočet jiných variant bezpečných databázových serverů. Důvodem volby MySQL je jeho snadná správa a také nabídka bezplatných řešení. Databázový server dokáže zvládat obsluhu poměrně velkého náporu dotazů (samozřejmě záleží také na hardwarové specifikaci serveru), je možné jej snadno zálohovat a v případě nutnosti je možné jej snadno přenést na jiné zařízení. V navrženém přístupovém systému používám databázový server pro management uživatelů systému a také jako logovací server přístupů. Pokud by se jednalo o opravdu rozsáhlý přístupový systém, bylo by pravděpodobně vhodné provozovat dva oddělené databázové servery pro uživatele a logování.

5.2.1 Autentizační applet pro platformu Java Card

Jak bylo zmíněno, applet pro čipovou kartu je hlavní částí celého systému. Na čipové kartě jsou implementovány metody pro správné vykonání PACE protokolu, dále je zde obsaženo zabezpečení appletu proti nežádoucím útokům a v neposlední řadě applet zprostředkovává funkci pro získání ID držitele karty po úspěšném vykonání PACE protokolu. V navrženém appletu je využito několika kryptografických prostředků:

- Šifrování **AES-CBC** o délce klíče 128b bez zarovnání
- hašovací funkce **SHA-1**
- **ECDH** algoritmus nad křivkami typu F_p nebo F_{2m}
- Podpis **MAC8-DES** o délce klíče 64b se zarovnáním ISO9797-M2 [16]
- Násobení velkých čísel metodou **Comba**

Pro usnadnění práce s eliptickými křivkami a také snadnou modifikovatelnost appletu jsem kromě hlavní třídy appletu vytvořil jednu pomocnou třídu `ECCurveSpec`, jejíž detailní popis metod a parametrů třídy ECCurveSpec je k dispozici v dokumentaci JavaDOC na přiloženém CD. Pomocí této třídy je definováno s jakým typem křivky karta pracuje a poté jsou nastaveny správné délky polí pro alokaci a kontrolu jejich délek při komunikaci s terminálem. Třída je nachystána pro použití všech doposud podporovaných typů křivek na platformě Java Card.

Parametry appletu, typ křivky a překlad appletu

Připraveny jsou dvě varianty appletu pro použití na čipových kartách. Podle toho, jaké jsou dispozice karty, na kterou chceme applet instalovat, je možné si vybrat buď applet `PACEDHApplet-KeyBuilder`, využívající předpřipravených křivek generovaných na kartě a nebo využít appletu `PACEDHApplet-CustomEC`, který používá

[1] Bezplatný vývojářský kit obsahuje vše co je zapotřebí pro provoz a správu MySQL databáze: `https://dev.mysql.com/downloads/installer/`

vlastních parametrů a klíčů pro eliptickou křivku. V obou případech je potřeba před překladem specifikovat to, s jakým druhem křivky bude karta pracovat, pomocí nachystaných definic ve třídě `ECCurveSpec`, dále s jakou délkou ID karta pracuje (musí být dělitelné 8) a posledním parametrem je délka náhodné veličiny, která se generuje v prvním kroku PACE protokolu. V základním nastavení pracuji s křivkou Fp 192b, délka ID 8B a délka náh. veličiny je 16B.

Dalším krokem je překlad appletu pomocí správné verze JCAPI, v závislosti na tom, jaké JCAPI podporuje karta, na kterou je applet cílen. O tomto tématu do detailu popisuje kapitola 2.2.5, takže ji zde nebudu více rozebírat. Po úspěšném překladu appletu máme již připravený .cap soubor pro instalaci na kartu.

Instalace na kartu a instalační parametry

Pro instalaci appletu na kartu je buď možné využít administrátorské terminálové aplikace, která kromě samotné instalace appletu provede i registraci uživatele, nebo je možné applet instalovat ručně například pomocí aplikace GlobalPlatform. Jak pracovat s gp.exe je vysvětleno podrobněji v kapitole 2.2.4. Při instalaci appletu se kartě posílá jak instalovaný .cap soubor, tak také specifické instalační parametry pomocí příkazu -params, které jsou povinné. Jedná se o předsdílené heslo PI a ID karty (uživatele). Bez těchto dvou parametrů neproběhne instalace appletu, je nutné dodržet správnou délku (definovanou při překladu appletu) a taktéž se kontroluje, zda tyto parametry nejsou nulové. Příkaz pro instalaci appletu, přeloženou pomocí JCAPI 2.2.2, na kartu se zvoleným předsdíleným heslem ve tvaru $d4afb74fe2f6bf380689ea532cc817c0_{16}$ a uživatelským ID $ffeeddccbbaa9988_{16}$, by vypadal následovně:

```
gp.exe -install PACEDHAppletJC222.cap -params d4afb74fe2f6bf380689ea532cc817c0ffeeddccbbaa9988
```

Při instalaci appletu se uloží předané parametry do paměti karty, alokují se správné velikosti používaných polí a taktéž se inicializují všechny třídy se kterými applet pracuje.

Důležité metody appletu a komunikace s terminálem

V kapitole 4.1 jsou popsány detaily protokolu PACE a je zde rozdělen na 5 dílčích částí. V appletu je protokol zredukován do 4 částí tím, že první a druhá část PACE protokolu je prováděna v jedné metodě `stepOne()` - hašování předsdíleného hesla, příjem kryptogramu z terminálu, generace klíčů eliptické křivky a odeslání veřejného klíče na terminál. Další metody pak odpovídají krokům protokolu PACE:

- `stepTwo()` - Třetí část PACE protokolu
- `stepThree()` - Čtvrtá část PACE protokolu
- `stepFour()` - Pátá část PACE protokolu

Applet využívá 4. varianty APDU příkazů popsaných v 2.1.3, tedy obousměrný přenos dat jedním řídícím příkazem, pro minimalizaci režie spojené s komunikací mezi kartou a terminálem. Pokud úspěšně proběhnou jednotlivé kroky PACE algoritmu – tj. karta v posledním kroku obdrží správný MAC podpis z terminálu - je povolen přístup k poslední metodě `getCardID()`, která zašifruje ID karty pomocí AES a odešle jej na terminál. Detailní popis jednotlivých funkcí je k dispozici v dokumentaci JavaDOC na přiloženém CD, stejně tak jsou zdrojové kódy řádně okomentovány pro pochopení činnosti.

5.2.2 Uživatelská terminálová aplikace přístupového systému

Terminálová aplikace je implementována v grafickém rozhraní JavaFX a její funkčnost je navržena jako konkurenčí vícevláknová aplikace. Kromě podpůrných tříd grafického ovladače JavaFX a dalších drobných tříd, se to hlavní odehrává ve 4 třídách, jejichž funkčnost bych chtěl více přiblížit. Jsou to třídy Database, PACE, Worker a FXMLDocumentController.

Třída Database

Tato třída zajišťuje veškerou komunikaci se databázovým serverem. Pomocí této třídy je navazováno zabezpečené spojení, kladeny SQL dotazy na databázi uživatelů a také odesíláno logování činnosti terminálové aplikace. Protože standardní JDK podporuje pouze základní komunikace s SQL databázemi pomocí tzv. Java Database Connectivity (dále jen JDBC), využívá tato třída externího konektoru (knihovny) **MySQL JDBC Driver**[2], která obsahuje všechny potřebné Java funkce pro správnou komunikaci s MySQL serverem. Popis funkčnosti jednotlivých metod je k dispozici v dokumentaci JavaDOC na přiloženém CD. Samotné SQL dotazy nejsou tvořeny jako obyčejné řetězce, ale pomocí třídy `PreparedStatement`[3] a to z důvodu ochrany proti útokům SQL injection.

Třída PACE

Tato třída je stěžejní pro vykonání celé funkčnosti protokolu PACE. Podobně jako v navrženém appletu obsahuje celkem 4 hlavní metody jednotlivých kroků PACE protokolu + další obslužné a doplňkové metody. Při nové instanci třídy je třídě nastaveno s jakým druhem křivek bude pracovat, je předán ukazatel na vybrané čtecí zařízení a dojde ke kompletní inicializaci všech potřebných kryptografických

[2] JDBC konektor a další typy driverů pro jiné programovací jazyky je možné nalézt na: `https://www.mysql.com/products/connector/`

[3] Více informací k třídě PreparedStatement a jejímu použití je možné nalézt na: `https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html`

prostředků a používaných polí. Vše ostatní už je v automatické režii této třídy, tudíž není nutné nastavovat další parametry. Voláním jedné jediné funkce `doPace()` proběhne kompletní PACE protokol a získání ID karty. Samozřejmostí je kontrola možných chyb v průběhu protokolu, což znamená, že tato funkce vrací unikátní návratovou hodnotu, která specifikuje zda došlo k nějaké chybě (případně jaké) či nikoliv. Jednotlivé návratové hodnoty, popis funkčnosti metod a použitých parametrů je k dispozici v dokumentaci JavaDOC na přiloženém CD.

Protože PACE protokol pracuje s eliptickými křivkami, tak i tato třída s nimi musí korektně pracovat. Nativně JDK sice do jisté míry práci s eliptickými křivkami podporuje, ale bohužel jen okrajově a s omezenými možnostmi. Metody jsou dosti neohrabané a občas chybí i některé důležité možnosti nastavení pro křivky. Protože v navrženém systému pracuji hlavně s křivkami generovanými na kartách (tedy křivky dle SEC2v1), zvolil jsem alternativní volně použitelnou knihovnu pro práci s eliptickými křivkami. Jedná se o multiplatformní sbírku knihoven s názvem **Bouncy Castle**[4] (dále jen BC), která kromě eliptických křivek disponuje mnoha dalšími kryptografickými a bezpečnostními třídami a metodami. Kromě většího množství možností nastavení parametrů křivek se také dá využít už definovaných křivek dle jejich standardizovaného pojmenování. Díky tomu bylo možné velmi snadno pracovat s křivkami SEC2, protože BC je má uložené a k načtení všech potřebných parametrů stačilo zvolit požadovanou křivku pomocí jejího názvu, tzn. například pro křivku typu F_p o délce klíče128b označenou v dokumentu SEC2v1 jako **secp128r1** by příkaz pro načtení parametrů křivky vypadal takto:

```
ECNamedCurveTable.getParameterSpec("secp128r1");
```

Díky této knihovně se tak velice usnadnila následná práce s křivkami a algoritmem ECDH. Jak je zmíněno v sekci 3.2.2, čipová karta na rozdíl od terminálové aplikace provádí dodatečně ještě hašování výsledného sdíleného klíče ECDH algoritmu, proto bylo nutné pro správné provedení PACE protokolu provádět hašování i na terminálové aplikaci.

Třída Worker

Celá tato třída je implementována jako samostatné vlákno aplikace a implementuje celý chod systému popsaný na obr. 5.2. Po spuštění zajišťuje neustálé sledování vybraného čtecího zařízení a čeká na vložení karty do čtečky. Po vložení karty se jako první dotáže pomocí vyskakovacího okna na počáteční heslo a v případě správného zadání pokračuje volbou appletu na vložené kartě, následným zpracováním PACE protokolu a v případě úspěšného dokončení a získání ID také dotázání se databáze na přístupová práva uživatele. Samozřejmostí je logování případných chyb do databáze.

[4] Více o Bouncy Castle na oficiálních stránkách: `https://www.bouncycastle.org/`

Po jednom takovém autentizačním cyklu vyčkává na vyjmutí karty ze čtečky, tím je celý cyklus restartován od začátku a je možné opět přiložit kartu do čtecího zařízení.

Třída Worker je implementována jako nekonečná smyčka, která se automaticky spouští na terminálu jako samostatné vlákno po jejím instancování. Pokud toto vlákno aktuálně běží, není možné měnit parametry zvoleného databázového serveru ani měnit vybrané čtecí zařízení. Aby tyto dvě věci bylo možné změnit, je zapotřebí vlákno voláním metody `stopWorker()` zastavit. Popis funkčnosti metod a použitých parametrů je k dispozici v dokumentaci JavaDOC na přiloženém CD.

Třída FXMLDocumentController

Hlavní řídící třída platformy JavaFX, který zajišťuje správné zobrazování celého uživatelského rozhraní (dále jen GUI). Obsluhuje všechny aktivní prvky v GUI a vykresluje veškeré změny na obrazovce. Tato třída má na starosti hlavně zmíněné GUI a také počáteční inicializaci celé aplikace, jako například zjištění přítomnosti čtecích zařízení v systému a kontrolu zadávaných údajů. Také je zajišťuje přepínání mezi jazykovými verzemi aplikace a správně nastavení všech přeložených textů mezi českou a anglickou verzí. V případě možných chyb nebo varovných oznámení informuje uživatele o druhu problému.

5.2.3 Správcovská terminálová aplikace přístupového systému

Tato aplikace je pouze pro použití správcem systému a je rozdělena na část autentizační a část registrační. Celá aplikace je navržena hlavně pro potřeby správce – disponuje širšími možnostmi nastavení pro ověřování správně funkčnosti karet, podrobnější výpis prováděných akcí a hlavně je díky této aplikaci možné instalovat applety na čipové karty včetně registrace nových uživatelů do systému. Protože autentizační část aplikace je až na pár drobných detailů v názvech tříd a metod totožná s uživatelskou terminálovou aplikací, zmíním se pouze o registrační části aplikace. I zde nechybí přepínání mezi jazykovými verzemi aplikace.

Aby nemusela být instalace appletů a registrace uživatelů prováděna manuálně, je správcovská aplikace vybavena jednoduchým nástrojem pro obsluhu této činnosti. Je k tomu zapotřebí pouze vložená čipová karta do čtecího zařízení, navázané spojení s databázovým serverem a vybrané správné čtecí zařízení. Pokud jsou všechny tyto podmínky splněny, je možné vyplnit všechny údaje o novém uživateli. Dále je potřeba vědět parametry vydávané čipové karty a podle toho zvolit druh použité eliptické křivky a také verzi JCAPI, které karta podporuje. Protože pro instalaci appletů na kartu je využíváno aplikace GlobalPlatform (gp.exe), je nutné aby v adresáři

odkud je správcovská aplikace spouštěna existovala podsložka applet, obsahující jak gp.exe tak samotné .cap soubory příslušných appletů. Takto prováděná instalace je možná pouze pro čipové karty podporující generování klíčů přímo na kartě (např. NXP JCOP). Pokud je požadováno využití vlastních parametrů křivek, je instalace appletu nutné provést ručně, stejně tak registraci uživatele do databáze.

5.2.4 Databázová aplikace

Tato aplikace je nástrojem pro zobrazování dat z databáze uživatelů a logovací databáze. Podobně jako v uživatelské aplikaci je zde hlavní částí třída Database zajišťující přípojení a dotazy na databázi. Funkčnost třídy je totožná, takže ji zde nebudu už znovu rozebírat a zaměřím se spíše na možnosti této aplikace. V levé části je možné provádět dotazy nad databází uživatelů, tedy vyhledávání dle různých kritérií. Pravá část potom slouží na dotazy nad logovací databází, jako například zobrazování logů v daném časovém úseku, zobrazování logů daného uživatele nebo daného terminálu. Protože hlavním cílem této práce byla hlavně bezpečnostní část, je funkčnost této zobrazovací aplikace velmi jednoduchá a určitě by byla vhodným kandidátem pro rozšíření o další dotazy, možnost kombinovat více kritérií do jednoho vyhledávání a podobně. I zde je možnost volby přepnutí mezi jazykovými verzemi.

5.2.5 Databázový MySQL server

Jako databázi jsem použil volně dostupný server MySQL. Vývojářský instalační balíček obsahuje vše potřebné pro chod a správu databáze - samotný server, grafické rozhraní pro úpravu nastavení a editaci databází, správu uživatelů databáze a také monitorovací aplikace. Po instalaci jsem v nastavení serveru povolil (a vynutil) používání SSL/TLS zabezpečení pro veškerou komunikaci se serverem a následně jsem vytvořil tabulky, kterých přístupový systém bude využívat. Tabulky jsou aktuálně vytvořeny dvě, jedna jako databáze uživatelů a druhá jako logovací databáze přístupů a chyb. Pro tvorbu těchto tabulek jsem použil následující SQL příkazy:

Vytvoření tabulky uživatelů

```
CREATE TABLE 'users' (
 'id' varchar(16) NOT NULL,
 'gid' int(11) NOT NULL,
 'name' varchar(45) NOT NULL,
 'surname' varchar(45) NOT NULL,
PRIMARY KEY ('id'))
ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Vytvoření logovací tabulky

```
CREATE TABLE 'accesslog' (
'index' int(11) NOT NULL AUTO_INCREMENT,
'cardid' varchar(16) NOT NULL,
'location' varchar(45) NOT NULL,
'grantstatus' bit(1) NOT NULL,
'date' datetime NOT NULL,
PRIMARY KEY ('index'))
ENGINE=InnoDB AUTO_INCREMENT=8660 DEFAULT CHARSET=utf8;
```

Tabulka uživatelů uchovává ID uživatele, jeho jméno a příjemní a také přístupovou skupinu. Primárním klíčem tabulky je ID uživatele a žádné z polí nesmí být prázdné. Logovací tabulka nese informace o ID uživatele který přistupoval, v případě logování nějaké chyby je toto pole využito pro zapsání druhu chyby. Dále pak pole udávající lokalitu, ze které bylo logování (tzn název terminálu) provedeno, booleovskou hodnotu zda byl udělen nebo zamítnut přístup a posledním polem této tabulky je časové razítko, kdy k události došlo. Tabulka je také opatřena vlastním indexem, který je zde pouze pro potřeby primárního klíče tabulky, není s ním však nijak zacházeno a jeho hodnota je automatickým inkrementováním určována přímo databázovým serverem.

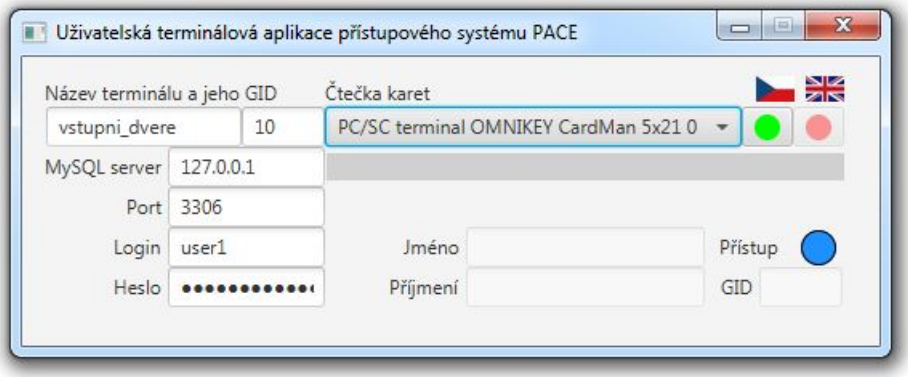
5.3 Popis GUI a ovládání aplikací

Tato podkapitola je spíše uživatelským manuálem k použití jednotlivých aplikací. Pomocí přiložených snímků z aplikací bych rád popsal jednotlivé prvky aplikací. Aplikace které jsem implementoval jsou připraveny ve dvou jazykových verzích, které jsou však funkčně naprosto stejné, takže se budu věnovat popisům pouze českých verzí. Přepínání jazykových verzí je ve všech aplikacích umístěno poblíž pravého horního rohu a funkčnost není potřeba nijak zvlášť popisovat.

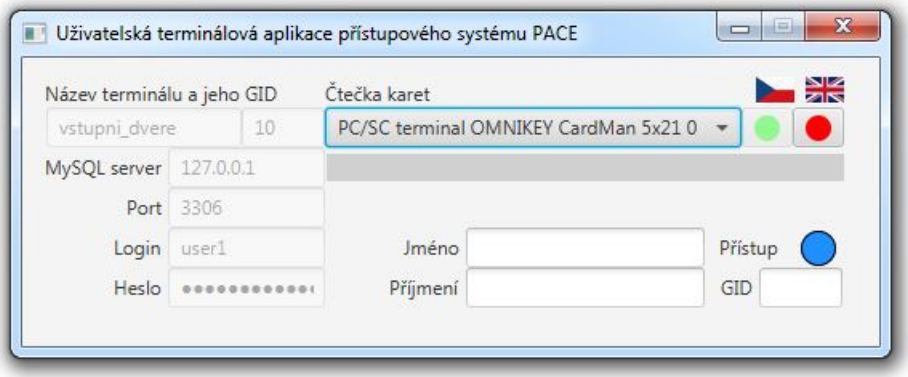
5.3.1 Uživatelská terminálová aplikace

Po spuštění je k dispozici po levé straně aplikace několik polí pro nastavení aplikace. Hlavními prvky jsou pojmenování terminálu a určení jeho GID. Tyto dva parametry se budou následně promítat do logovací databáze a GID určuje přístupovou skupinu daného terminálu. Tato dvě pole nesmí být prázdná a GID musí být kladné nenulové číslo. Dále se zde nachází 4 pole pro nastavení databáze, tedy adresa, port, přihlašovací jméno a heslo. Taktéž tato dvě pole nesmí být prázdná. Posledními aktivními prvky jsou už jen výběrové menu čtecího zařízení a zelené tlačítko. Po vyplnění

všech údajů stačí ve výběrovém menu vybrat čtecí zařízení, které chceme používat. Stisknutím zeleného tlačítka se ověří vyplněné údaje, naváže se spojení s databází a aktivuje se smyčka čekající na vložení čipové karty do čtečky. Pokud se nebude dát připojit k databázi nebo je některé z polí vyplněno špatně, je o tom uživatel informován ve stavovém pruhu pod výběrovým menu čtečky. Pokud je vše v pořádku a spustí se čekací smyčka, není možné již za chodu upravovat údaje, protože dojde k uzamknutí polí, což znázorňuje obr. 5.3. Údaje lze znovu upravit pouze po ručním zastavení chodu aplikace červeným tlačítkem. Rozvržení aplikace před startem čekací smyčky je znázorněno na obr. 5.4.



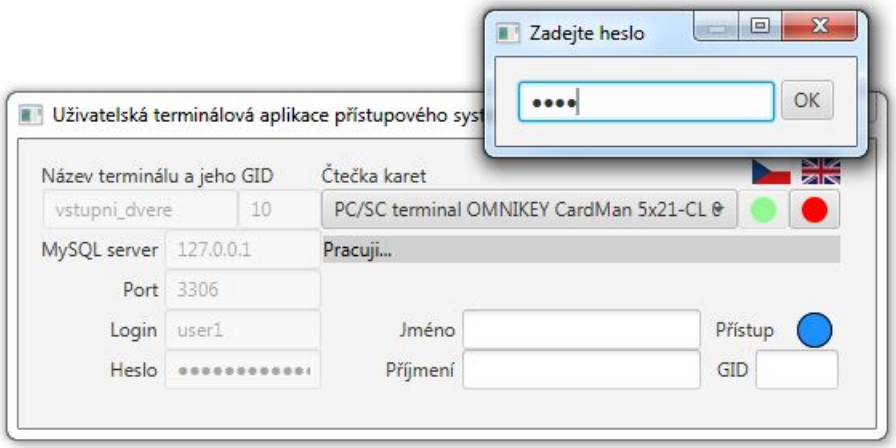
Obr. 5.3: GUI uživatelské aplikace - nastavení parametrů



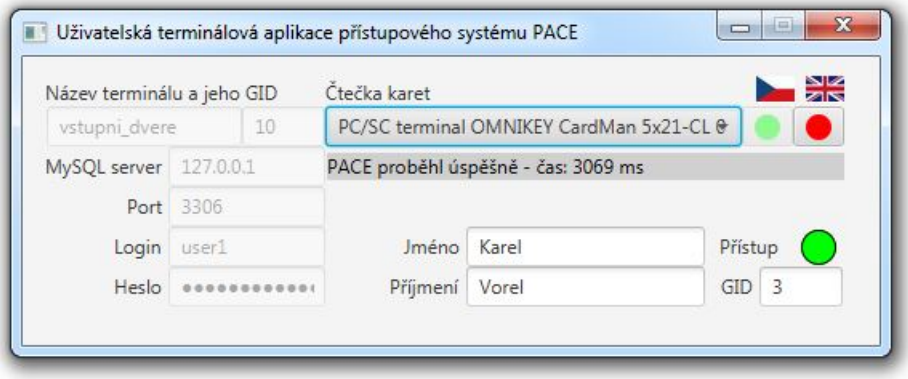
Obr. 5.4: GUI uživatelské aplikace - spuštění čekací smyčky

Když dojde k vložení čipové karty do čtečky, jako první vyskočí nové okno dotazující se na počáteční heslo, jak ukazuje obr. 5.5. Pokud je heslo vyplněno špatně, je o tom uživatel informován a je dotázán na heslo znovu. Po zadání správného hesla proběhne autentizační proces včetně zalogování do databáze a výsledek tohoto procesu je zobrazen. V případě chyby se vypíše a zaloguje příslušná chybová hláška. Po

načtení ID z karty je zobrazeno info o uživateli a je rozhodnuto, zda tomuto uživateli bude nebo nebude udělen přístup. Výsledek po autentizačním procesu zobrazuje obr. 5.6.



Obr. 5.5: GUI uživatelské aplikace - dotaz na počáteční heslo

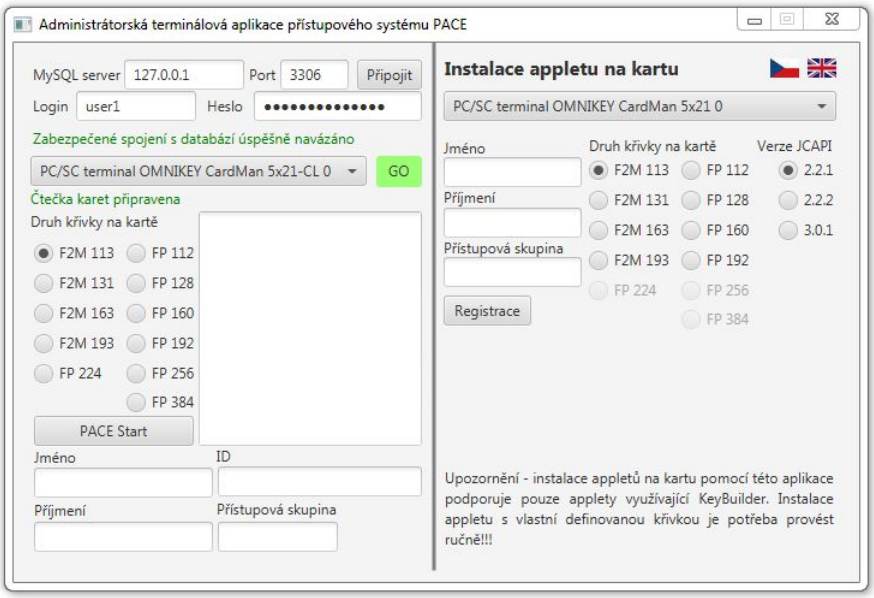


Obr. 5.6: GUI uživatelské aplikace - zobrazení výsledku autentizace

5.3.2 Správcovská terminálová aplikace

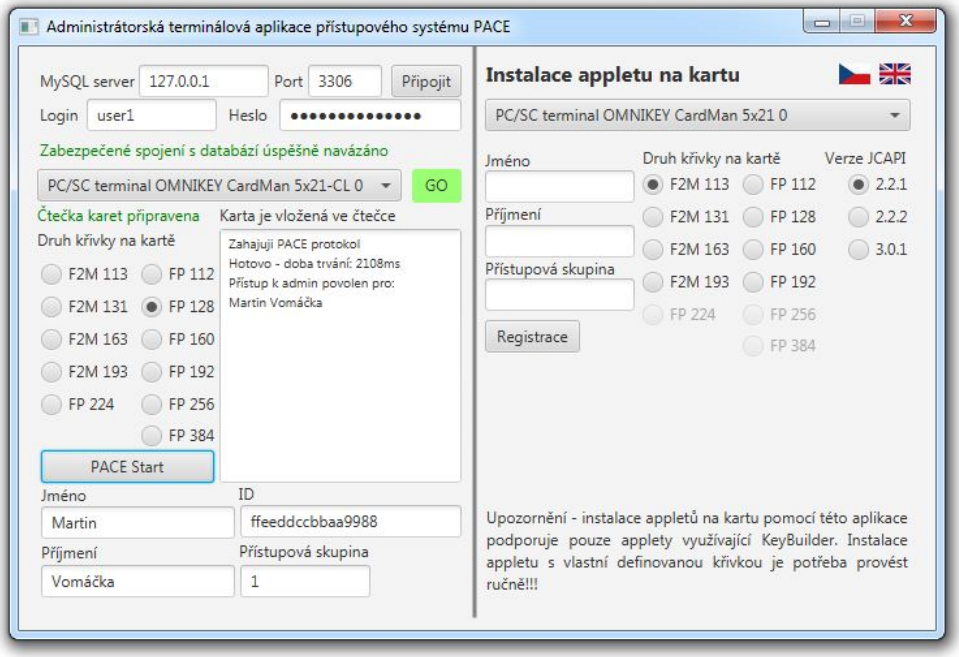
Jak bylo řečeno v popisu implementace, tato aplikace je spíše nástrojem pro ověřování funkčnosti karet správcem systému a registraci uživatelů do systému. Je rozdělena na levou část, která má na starost připojení k databázi a jsou zde ovládací prvky pro autentizaci, a na pravou část, která slouží k instalaci appletů na kartu. Stejně jako v uživatelské aplikaci jsou zde 4 pole pro nastavení údajů pro databázi,

dále výběrové menu čtecího zařízení. Po vyplnění správných údajů databáze, úspěšném připojení po stisku tlačítka **Připojit** a vybrání požadované čtečky se stejně jako u uživatelské aplikace, po stisknutí tlačítka **GO**, spustí čekací smyčka - zelená barva indikuje, že je spuštěná. Zde však čekací smyčka nezačne automaticky po vložení karty provádět autentizaci, ale pouze o vložení karty informuje ve stavovém řádku. Uživatelské rozhraní po spuštění čekací smyčky je zobrazeno na obr. 5.7. Správce má pak možnost si vybrat, s jakým druhem eliptické křivky bude následná autentizace pracovat a její provedení se stiskne až následovným stiskem tlačítka **PACE Start**. Proběhne autentizace a dotaz na databázový server a všechny prováděné kroky jsou vypisovány v pravém sloupci vedle tlačítka. Po úspěšné autentizaci jsou také zobrazeny informace o držiteli karty, jak ukazuje obr. 5.8.

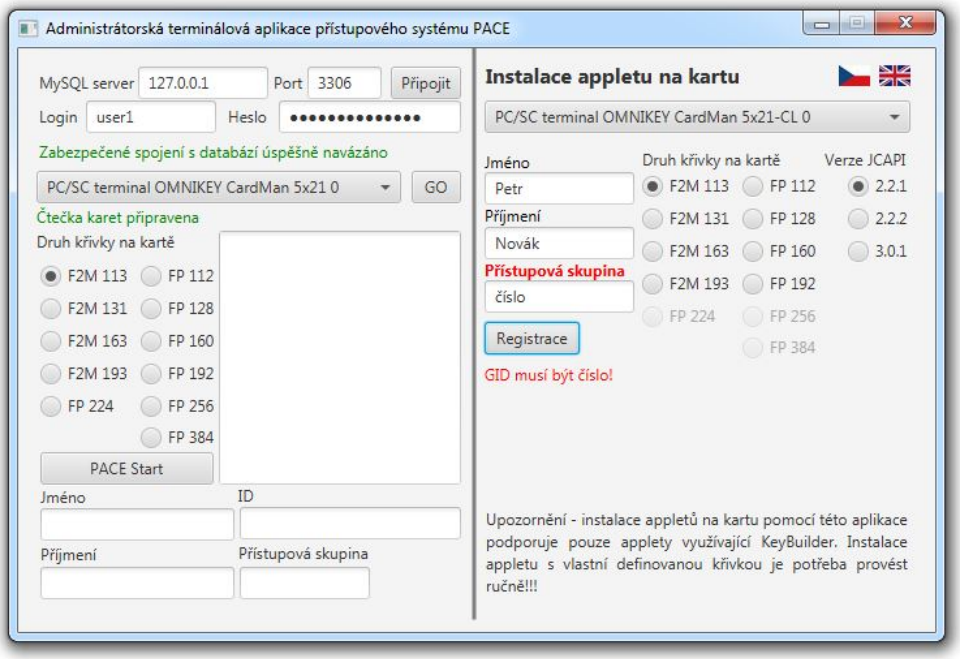


Obr. 5.7: GUI správcovské aplikace - spuštění čekací smyčky

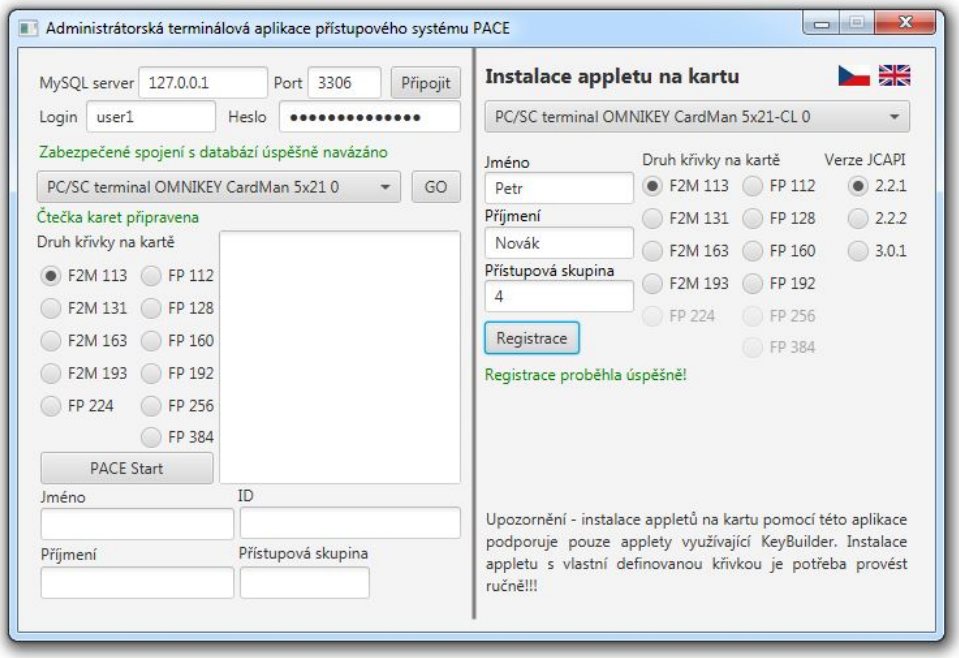
Pravá část aplikace je věnována instalaci appletů na kartu. Opět je zde možné vidět výběrové menu čtecího zařízení a dále je zde několik skupin radio tlačítek, kterými je možné navolit druh křivky, který chceme na kartu instalovat a také verzi JCAPI, kterým karta disponuje. Dále se zde nachází 3 pole pro vyplnění údajů o novém uživateli, která je povinné vyplnit – v případě nevyplnění není možné pokračovat v instalaci a o tomto faktu je správce náležitě informován chybovou hláškou, jak ukazuje obr. 5.9. Pokud je vše vyplněno v pořádku, stiskem tlačítka **Registrace** proběhne generace nového ID, zápis všech údajů do databáze uživatelů a nakonec proběhne také instalace appletu na kartu, což zobrazuje obr. 5.10. Instalace je spouštěna jako samostatný proces systému, proto je potřeba pár sekund setrvat, než instalace úspěšně proběhne (lze poznat také tak, že přestane blikat čtečka karet).



Obr. 5.8: GUI správcovské aplikace - zobrazení výsledku autentizace



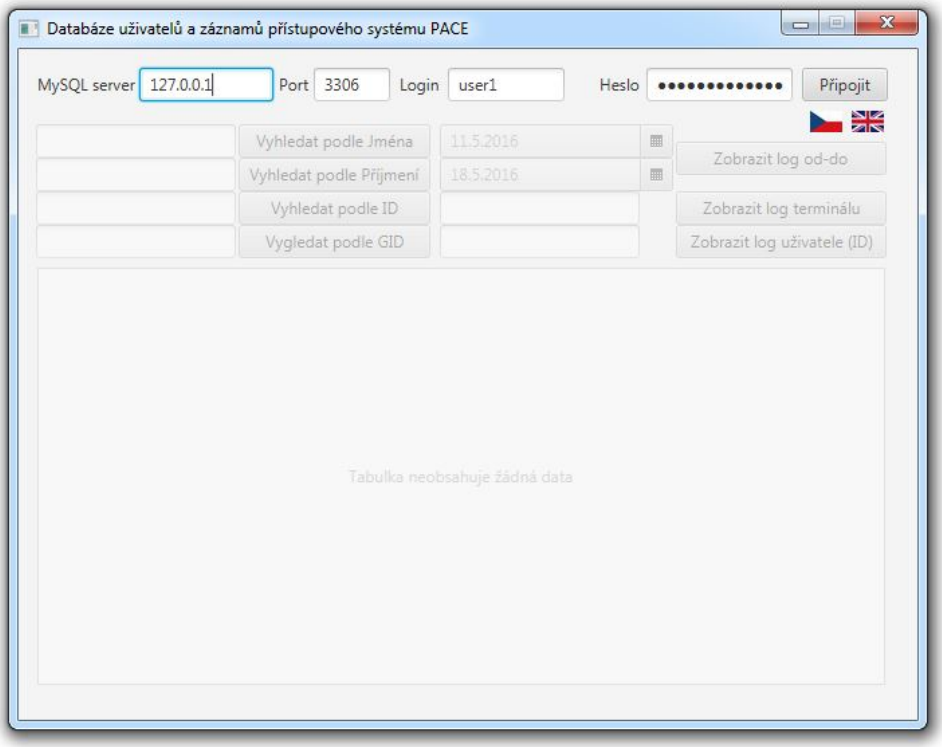
Obr. 5.9: GUI správcovské aplikace - kontrola správnosti vyplněných údajů



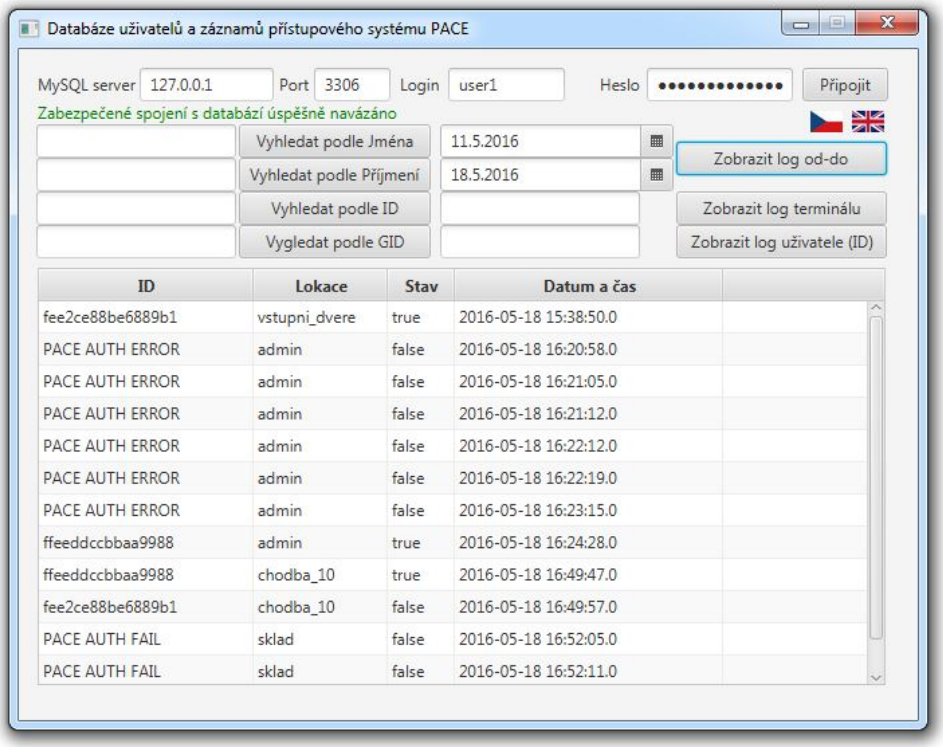
Obr. 5.10: GUI správcovské aplikace - registrace uživatele a instalace appletu

5.3.3 Databázová aplikace

Tato aplikace je nejvíce intuitivní na použití. Opět jako v předchozích případech jsou zde pole pro údaje databáze a tlačítko pro připojení k databázi. Pokud neproběhne úspěšné připojení, je o tom uživatel informován ve stavovém řádku a nedojde k odemčení zbytku funkčnosti aplikace. Obr. 5.11 ukazuje GUI aplikace po jejím spuštění. Po úspěšném připojení je zbytek aplikace odblokován a lze používat ovládací prvky. Sada polí a tlačítek v levém sloupci slouží pro vyhledávání v databázi uživatelů, ta v pravém sloupci zase pro vyhledávání údajů v logovací databázi. Vyhledání je možné pouze pro jednotlivá pole, nelze kombinovat více vyhledávacích kritérií (jako například vyhledávat podle jména a příjmení). Veškeré načtené výsledky se zobrazují v tabulce pod ovládacími prvky. Pokud se po stisku tlačítka nezobrazí žádné údaje, znamená to, že databáze nenašla žádnou shodu s vyhledávaným výrazem. Výsledek zobrazení logů v rozsahu jednoho týdne ukazuje obr. 5.12.



Obr. 5.11: GUI databázové aplikace - aplikace po spuštění



Obr. 5.12: GUI databázové aplikace - zobrazení logů v rozsahu jednoho týdne

5.4 Časová analýza autentizačního systému

Rychlost vykonání autentizačního procesu je do jisté míry také důležitým prvkem přístupového systému. Samozřejmě záleží na prostředí do kterého je takový systém nasazen a v jaké souvislosti se autentizace provádí. Může se jednat například o přístupy k fyzickým prostorám (budovy, poschodí, místnosti), kde hraje rychlost autentizace vyšší roli než například u přístupů k logickým objektům (přihlašování na pracovní stanici, platby a dobíjení kreditu na kartě). U přistupování k fyzickým prostorám je pochopitelné, že autentizace by měla být rychlá – nikomu se jistě nechce čekat několik vteřin, než bude vpuštěn přes zamčené dveře, obzvlášť pak v případě, kdy takový úkon provádí mnohokrát za den. Naopak pokud je karta a autentizace pomocí ní použita například na pracovní stanici stolního počítače, kde kartu vložíme do čtečky a můžeme ji tam buď nechat celou dobu nebo jen po dobu spouštění systému, už tolik nezáleží, jestli proces autentizace bude trvat 1 nebo 10 sekund. Obvykle spuštění stolního počítače trvá i několik desítek vteřin a samotný proces autentizace je možné provést už při načítání.

Původní autentizační schéma navržené v semestrální práci pracovalo s velmi jednoduchým principem autentizace a díky tomu byl velice rychlý – časy celé autentizace se pohybovaly okolo 200ms. Docházelo pouze k vygenerování časového razítka na terminálu, jeho zašifrování a odeslání na kartu, kde bylo toto razítko logickou operací XOR sečteno s ID karty, opět zašifrováno a zasláno zpět na terminál. Takový systém by se určitě hodil jako přístupový systém k fyzickým objektům, právě díky své rychlosti. Nově navržené schéma je o poznání složitější, už jen co se počtu kryptografických operací týká. Protože časy operací v rámci terminálové aplikace jsou zanedbatelné, budeme-li brát v potaz jeho provoz na adekvátně výkonném zařízení, zmíním se hlavně o časech jednotlivých operací na čipové kartě. Na čipové kartě dochází celkem k těmto operacím:

- 4x hašování SHA-1
- 2x šifrování AES
- 1x generace klíčů nad EC
- 1x ECDH algoritmus
- 3x násobení metodou Comba
- 1x sčítání velkých čísel
- 2x MAC podepisování
- 5x obousměrný přenos dat

V tabulce 5.1 jsou uvedeny časy jednotlivých fází protokolu PACE na dvou kartách. U dalších dvou JCOP karet chyběla podpora AES šifry. Z tabulky je patrné že nejvíce časově náročný je třetí krok. To hlavně z důvodu násobení velkých čísel (násobení dvou polí o délce 96B) metodou Comba. Java karty jsou v tomto ohledu ještě více znevýhodněny oproti například kartám .NET, protože pracují maximálně s číselným datovým typem **`short`** na rozdíl od **`int`** na .NET kartách. Taktéž druhá část

PACE protokolu zabírá relativně dlouhou dobu. Sice se zde při násobení Comba pracuje s ještě relativně krátkými čísly, ale také zde probíhá ještě generace sdíleného klíče ECDH protokolu. Čas odeslání zašifrovaného ID karty na terminál je v porovnání se zbytkem protokolu zanedbatelný, u obou měřených karet se pohyboval v desítkách milisekund. Časová náročnost úkonů prováděných na kartě je také dost zredukována inicializaci veškerých používaných tříd a polí při instalaci appletu.

	NXP JCOP J3A040 CL	**SmartCafe Expert 6.0**
První část PACE algoritmu: 1) Zhašuje předsdílené heslo. 2) Vytvoří klíč pro dešifrování nonce. 3) Dešifruje přijatou zprávu a získá nonce. 4) Vygeneruje veřejný a soukromý ECDH klíč karty. 5) Získá bajtovou reprezentaci bodu veřejného klíče. 6) Získá bajtovou reprezentaci generátoru G. 7) Inicializuje KeyAgreement pomocí soukromého ECDH klíče karty. 8) Odešle veřejný ECDH klíč na PC.	312 ms	299 ms
Druhá část PACE algoritmu: 1) Vygeneruje sdílené tajemství z přijatého veřejného ECDH klíče PC. 2) Provede násobení a sčítání pro získání G' - dle PACE algoritmu. (Comba) 3) Vygeneruje soukromý DH klíč karty. 4) Vytvoří veřejný DH klíč karty. (Comba) 5) Odešle veřejný DH klíč na PC.	620 ms	450 ms
Třetí část PACE algoritmu: 1) Vygeneruje hlavní klíč z přijatého veřejného DH klíče PC. (Comba) 2) Tvorba klíčů K-ENC, K-MAC a K-'MAC pomocí hašovací funkce. 3) Vytvoření MAC podpisu. 4) Odeslání podpisu na PC.	1811 ms	1176 ms
Čtvrtá část PACE algoritmu: 1) Zkontroluje přijatý MAC podpis 2) Odešle na PC zda je podpis v pořádku nebo ne.	77 ms	42 ms
Celková doba provádění všech dílčích částí (bez komunikace s PC)	2875 ms	1963 ms
Celková doba provádění všech dílčích částí (včetně komunikace s PC)	2933 ms	2021 ms
Celková doba provádění všech dílčích částí + přenos ID karty	2997 ms	2053 ms

Tab. 5.1: Změřené časy vykonání jednotlivých fází PACE algoritmu

5.5 Bezpečnostní analýza autentizačního systému

V navrženém přístupovém systému mají na sebe jednotlivé kroky přímou návaznost. Bez úspěšného provedení kroků předchozích není možné pokračovat v dalších krocích. Prvním ochranným prvkem, který je v systému kontrolován, je počáteční heslo. Bez znalosti počátečního hesla, které má terminálová aplikace napevno uložené ve zdrojovém kódu, není možné vůbec zahájit jakoukoliv komunikaci s kartou nebo s databázovým serverem. Toto heslo si musí uživatelé pamatovat a heslo by mělo být adekvátně složité. Použití takového hesla bylo zvoleno jako ochrana před odcizením nebo ztrátou čipové karty. Po zadání správného hesla a volbě appletu na čipové kartě se zahajuje vykonání protokolu PACE.

Ochrana proti odposlechu

Čipová karta má v sobě uložené jen předsdílené heslo, parametry použité eliptické křivky a své ID. Kdyby se útočníkovi podařilo nějakým způsobem získat uživatelovo ID, stále potřebuje znát přesné parametry použité křivky a předsdílené heslo, pro správně vykonání PACE protokol a ustanovení hesla pro šifrování přenosu ID. Bez vykonání všech kroků terminál ignoruje jakoukoliv komunikaci týkající se přenosu ID karty. U všech kroků je při komunikaci APDU příkazy striktně kontrolována i jejich délka nebo zda nejsou zasílána nulová data. Délka předsdíleného hesla může být v podstatě libovolná, protože v prvním kroku PACE protokolu je toto heslo hašováno pro tvorbu klíče AES šifrování. Získáním předsdíleného hesla by se tak útočník dostal maximálně k přenášené náhodné veličině s, která je však použita až po provedení ECDH algoritmu. Tím pádem je útočník, kterému se povedlo získat předsdílené heslo, vystaven řešení problému diskrétního logaritmu nad eliptickými křivkami.

Ochrana proti útoku hrubou silou

Útočník by se také mohl pokusit o útok hrubou silou na samotnou čipovou kartu s cílem získat z komunikace nějaké informace. Zde je karta opatřena počítadlem špatných pokusů a po jeho vyčerpání je veškerá komunikace appletu zablokována. To znamená, že útok hrubou silou je na jednu čipovou kartu nereálný. Na opačné straně by mohl útočník zkoušet útok na samotný terminál. Protože komunikační rychlost čipových karet je používána nejčastěji 9600b/s, už jen hádání první překážky, tj. 128b klíče AES šifry, by probíhalo rychlostí 75klíčů/s. Při daném počtu kombinací 2^{128} to je přibližně $4,54 \cdot 10^{36} s \cong 1,44 \cdot 10^{29}$let. Dále by musel stejným způsobem prolomit 160b klíče ECDH algoritmu a následného DH algoritmu a na závěr ještě odeslat 64b MAC podpis.

Ochrana autentizace databázovým serverem

Posledním ochranným prvkem před rozhodováním o udělení nebo zamítnutí přístupu je komunikace s databázovým serverem, vyhledání údajů o uživateli s poskytnutým ID a zjištění jeho přístupové skupiny. Komunikace s databází je opět řízena z terminálové aplikace, takže pokud nejsou informace z databáze vyžádány, tak aplikace nijak nereaguje na jakoukoliv síťovou komunikaci. Navíc je veškerá komunikace se serverem zabezpečena šifrovanou komunikací SSL/TLS, která je v současné době považována za vysoce bezpečnou. Server MySQL databáze také podporuje práci s certifikáty. Pokud by se útočníkovi nějakým způsobem povedlo obejít ochranu SSL/TLS, mohl by se pokusit o vložení nového fiktivního uživatele do databáze, ale

pro tyto účely je zápis do databáze uživatelů povolen pouze správcovské aplikaci, ke které má přístup pouze důvěryhodná pověřená osoba. Uživatelská terminálová aplikace má pouze práva čtení z databáze uživatelů.

6 ZÁVĚR

V rámci diplomové práce jsem se detailně seznámil s čipovými kartami, konkrétně s platformou JavaCard. Seznámil jsem se s komunikačním protokolem a s výpočetními prostředky různých druhů Java karet a objasnil jsem si podmínky pro tvorbu aplikací pro tento druh čipových karet. Cílem práce bylo navrhnout přístupový systém využívající čipových karet pro autentizaci uživatelů, s využitím moderních metod kryptografie. Aby bylo možné navrhnout vhodné autentizační schéma, bylo nutné zvolit vhodnou platformu pro implementaci a také určit dostupné kryptografické prostředky, se kterými bude možné v návrhu počítat.

Jako základ celého autentizačního schématu byl zvolen protokol PACE, a to hlavně z důvodů kombinace několika různých algoritmů spojujících výhody symetrické i asymetrické kryptografie a využívajících eliptických křivek, které jsou na platformě Java Card k dispozici. Po analýze tohoto protokolu byl pak ještě navrhovaný autentizační systém doplněn o další prvky pro zvýšení bezpečnosti celého systému. Přibyla ochrana autentizace pomocí počátečního hesla, ochrana appletu na čipové kartě a také zajištění ochrany v rámci databáze uživatelů.

Implementací navrženého autentizačních schématu vznikly celkem dva druhy appletů pro platformu Java Card - jeden (hlavní applet) využívající předpřipravených eliptických křivek u čipových karet NXP JCOP a generace klíčů ECDH algoritmu přímo na kartě a také druhá verze appletu, ve které lze použít vlastních parametrů křivek definovaných ručně - tento applet byl využit například pro kartu SmartCafe 6.0, aby bylo možné obě verze appletů porovnat. Dále vznikly celkem tři aplikace pro nasazení na terminálových stanicích. Hlavní uživatelská aplikace, která slouží čistě k autentizaci a udělování přístupu, byla navržena pro snadné použití a přehlednost. Potom také dvě aplikace sloužící správci systému. Pomocí správcovské terminálové aplikace lze kromě testování appletů na čipových kartách také registrovat nové uživatele do systému včetně instalace appletů na nově vydávané karty. Druhou správcovskou aplikací lze pohodlně vyhledávat údaje v databázi uživatelů a v logovací databázi přístupů. Poslední částí implementace byl také návrh databáze uživatelů a databáze logování přístupů, kde jsem zvolil MySQL server, který disponuje zabezpečenou komunikací SSL/TLS.

Celý navržený systém byl řádně odladěn a otestován, snahou bylo pokrýt i možné chybové stavy a zajistit bezproblémový chod. Všechny možné chybové stavy byly řádně ošetřeny a v případě jejich výskytu je o nich uživatel řádně informován. Součástí testování bylo i měření doby potřebné k autentizaci uživatele, kde vyplynulo, že nejvíce času při autentizaci zaberou aritmetické operace násobení. Použita byla metoda násobení Comba, ale protože platforma Java Card disponuje pouze datovým typem `short`, zabralo násobení poměrně velkou část celkového času. Druhým

časově náročným krokem byly operace nad eliptickými křivkami, hlavně pak generování soukromého a veřejného klíče pro ECDH algoritmus.

Tato práce byla zaměřena hlavně na část implementace na platformě Java Card a také implementaci terminálových aplikací. Databázová část této práce by byla vhodným kandidátem o další rozšíření, modifikace i úpravy. To se týká jak samotné databáze, kde je prostor pro doplnění dalších tabulek, například kvalitnější systém přístupových skupin, tabulka zablokovaných nebo dočasně vyřazených ID, tak také databázové správcovské aplikace, kde by bylo vhodné doplnit možnosti vícepodmínkových vyhledávání v databází, použití sofistikovanějších a komplikovanějších SQL dotazů apod.

LITERATURA

[1] CHEN, Zhiqun. *Java Card technology for Smart Cards: architecture and programmer's guide.* Boston: Addison-Wesley, 2000, xxii, 368 s. Java series. ISBN 0201703297.

[2] HANSMANN, Uwe. *Smart Card Application Development Using Java.* Odpovědný redaktor Tomáš Hála. Druhé, přepracované vydání. New York: Springer, 2002, xvi, 305s. ISBN 3540432027.

[3] STALLINGS, William. *Cryptography and Network Security.* 4th edition. [s.l.] : [s.n.], 2006. 592 s. ISBN 0131873164.

[4] MENEZES, Alfred, VAN OORSCHOT, Paul, VANSTONE, Scott. *Handbook of applied cryptography.* Boca Raton: CRC Press, 1997. 780 s. ISBN 0849385237.

[5] KUMAR, Sandeep S. *Elliptic Curve Cryptography for Constrained Devices: Algorithms, Architectures, and Practical Implementations.* Saarbruecken, Germany: VDM Verlag, 2008. ISBN 9783639068597.

[6] CHEN, Zhiqun. *How to write a Java Card applet: A developer's guide.* [online]. In: JavaWorld. Jul 1, 1999.
Dostupné z:
`http://www.javaworld.com/article/2076450/client-side-java/how-to-write-a-java-card-applet--a-developer-s-guide.html`

[7] Standards for Efficient Cryptography *SEC 2: Recommended Elliptic Curve Domain Parameters.* [online]. Sep 20, 200.0 Version 1.0
Dostupné z: `http://www.secg.org/SEC2-Ver-1.0.pdf`l

[8] Federal Office for Information Security Germany. *Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS token.* Part 2 [online]. Germany: Feb 3, 2015.
Dostupné z:
`https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03110/index_htm.html`

[9] BENDER, Jens, FISCHLIN, Marc, KÜGLER, Dennis. *Security Analysis of the PACE Key-Agreement Protocol.* [online]. BSI, Germany: Dec 18, 2009.
Dostupné z: `https://eprint.iacr.org/2009/624.pdf`

[10] RFID portál. *Základní informace o technologii RFID* [online].
Dostupné z: `http://www.rfidportal.cz/index.php?page=rfid_obecne`.

[11] CardWerk Technologies. *Smart Card Technology* [online].
Dostupné z:
http://www.cardwerk.com/smartcards/smartcard_technology.aspx.

[12] PALJAK, Martin. *Building JavaCard applet CAP files with Ant* [online].
Dostupné z: https://github.com/martinpaljak/ant-javacard.

[13] PALJAK, Martin. *GlobalPlatformPro* [online].
Dostupné z: https://github.com/martinpaljak/GlobalPlatformPro.

[14] ŠVENDA, Petr. *JavaCard algorithms support test* [online].
Dostupné z: http://www.fi.muni.cz/~xsvenda/jcsupport.html

[15] ISO/IEC 7816-1 *Identification cards – Integrated circuit cards – Part 1: Cards with contacts – Physical characteristics* [online]. International Organization for Standardization, Geneva, Switzerland. 2011-02-15
Dostupné z: http://www.iso.org/iso/catalogue_detail?csnumber=54089

[16] ISO/IEC 9797-1 *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher* [online]. International Organization for Standardization, Geneva, Switzerland. 2011-03-01
Dostupné z: http://www.iso.org/iso/catalogue_detail?csnumber=50375

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

3DES	Triple DES
AES	Proprietary application identifier extension
AID	Application Identifier
APDU	Application Protocol Data Unit
ATR	Answer To Reset
CBC	Cipher Block Chaining
DES	Data (Digital) Encryption Standard
DH	Diffie Hellman algorithm
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman algorithm
EEPROM	Electrically Erasable Programmable Read-Only Memory
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
JCAPI	Java Card Application Programming Interface
JCOP	Java Card OpenPlatform
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
JDBC	Java Database Connectivity
MAC	Message Authentication Code
MD5	Message-Digest algorithm
NFC	Near Field Communication
NXP	NXP Semiconductors
OS	Operation System
PACE	Password Authenticated Communication Establishment
PIX	Proprietary application identifier extension
RAM	Random Access Memory
RFID	Radio-frequency identification
RID	National registered application provider
ROM	Read-Only Memory
RTF	Reader Talks First
SEC	Standards for Efficient Cryptography

SHA	Secure Hash Algorithm
SIM	Subscriber Identity Modulel
SQL	Structured Query Language
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TLDU	Transaction Protocol Data Unit
TLS	Transport Layer Security
TTF	Tag Talks First
USB	Universal Serial Bus
VoIP	Voice over IP
XML	Proprietary application identifier extension
XOR	Exclusive OR (exclusive disjunction)

SEZNAM PŘÍLOH

A PODPOROVANÉ ALGORITMY JAVA CARD

Následující čtyři tabulky zobrazují výsledky testů nástroje JCSupport test od Petra Švendy [14]. Jedná se o zredukovanou verzi výsledků, ve které nejsou zobrazeny ty řádky s algoritmy, které nebyly podporovány žádnou z šesti testovaných karet.

Kompletní data z výsledků testů je možné najít na přiloženém CD, ve složce jcsupport-results. Tyto soubory také obsahují (ve třetím sloupci) časové údaje, jak dlouho přibližně trvá provedení daného testovaného algoritmu na kartě.

Název karty	**JCOP 31 v2.2**	**JCOP 31 v2.3.2**	**JCOP J3A040**	**SC Exp. 3.2 144k**	**SC Exp. 4.X**	**SC Exp. 6.0**
	javacardx.crypto.Cipher					
DES_CBC_NOPAD	✓	✓	✓	✓	✓	✓
DES_CBC_ISO9797_M1	✓	✓	✓	✓	✓	✓
DES_CBC_ISO9797_M2	✓	✓	✓	✓	✓	✓
DES_CBC_PKCS5	✗	✗	✓	✓	✓	✓
DES_ECB_NOPAD	✓	✓	✓	✓	✓	✓
DES_ECB_ISO9797_M1	✓	✓	✓	✓	✓	✓
DES_ECB_ISO9797_M2	✓	✓	✓	✓	✓	✓
DES_ECB_PKCS5	✗	✗	✗	✓	✓	✓
RSA_ISO14888	✗	✗	✗	✓	✓	✓
RSA_PKCS1	✓	✓	✓	✓	✓	✓
RSA_NOPAD	✓	✓	✓	✓	✓	✓
AES_BLOCK_128_CBC_NOPAD	✓	✓	✓	✓	✗	✓
AES_BLOCK_128_ECB_NOPAD	✓	✓	✓	✓	✗	✓
RSA_PKCS1_OAEP	✗	✗	✗	✓	✓	✓
KOREAN_SEED_ECB_NOPAD	✗	✗	✓	✗	✗	✓
KOREAN_SEED_CBC_NOPAD	✗	✗	✓	✗	✗	✓
AES_CBC_ISO9797_M1	✗	✗	✗	✗	✗	✓
AES_CBC_ISO9797_M2	✗	✗	✗	✗	✗	✓
AES_CBC_PKCS5	✗	✗	✗	✗	✗	✓
AES_ECB_ISO9797_M1	✗	✗	✗	✗	✗	✓
AES_ECB_ISO9797_M2	✗	✗	✗	✗	✗	✓
AES_ECB_PKCS5	✗	✗	✗	✗	✗	✓
	javacard.crypto.Signature					
DES_MAC4_NOPAD	✗	✗	✗	✓	✓	✓
DES_MAC8_NOPAD	✓	✓	✓	✓	✓	✓
DES_MAC4_ISO9797_M1	✗	✗	✗	✓	✓	✓
DES_MAC8_ISO9797_M1	✓	✓	✓	✓	✓	✓
DES_MAC4_ISO9797_M2	✗	✗	✗	✓	✓	✓
DES_MAC8_ISO9797_M2	✓	✓	✓	✓	✓	✓
DES_MAC4_PKCS5	✗	✗	✗	✓	✓	✓
DES_MAC8_PKCS5	✗	✗	✗	✓	✓	✓
RSA_SHA_ISO9796	✓	✓	✓	✓	✓	✓
RSA_SHA_PKCS1	✓	✓	✓	✓	✓	✓
RSA_MD5_PKCS1	✓	✓	✓	✓	✓	✓
RSA_RIPEMD160_ISO9796	✗	✗	✗	✓	✓	✓
RSA_RIPEMD160_PKCS1	✗	✗	✗	✓	✓	✓
DSA_SHA	✗	✗	✗	✓	✗	✓
RSA_SHA_RFC2409	✗	✗	✗	✓	✓	✓
RSA_MD5_RFC2409	✗	✗	✗	✓	✓	✓
ECDSA_SHA	✓	✓	✓	✗	✗	✓

Tab. A.1: Přehled dostupných kryptografických prostředků na kartách - část 1.

Název karty	JCOP 31 v2.2	JCOP 31 v2.3.2	JCOP J3A040	SC Exp. 3.2 144k	SC Exp. 4.X	SC Exp. 6.0
	javacard.crypto.Signature					
AES_MAC_128_NOPAD	✓	✓	✓	✓	✗	✓
DES_MAC4_ISO9797_1_M2_ALG3	✗	✗	✗	✓	✓	✓
DES_MAC8_ISO9797_1_M2_ALG3	✓	✓	✓	✓	✓	✓
SEED_MAC_NOPAD	✗	✗	✓	✗	✗	✓
ECDSA_SHA_256	✗	✗	✗	✗	✗	✓
ECDSA_SHA_384	✗	✗	✗	✗	✗	✓
ECDSA_SHA_224	✗	✗	✗	✗	✗	✓
ECDSA_SHA_512	✗	✗	✗	✗	✗	✓
RSA_SHA_224_PKCS1	✗	✗	✗	✗	✗	✓
RSA_SHA_256_PKCS1	✗	✗	✗	✗	✗	✓
RSA_SHA_384_PKCS1	✗	✗	✗	✗	✗	✓
RSA_SHA_512_PKCS1	✗	✗	✗	✗	✗	✓
	javacard.security.RandomData					
PSEUDO_RANDOM	✓	✓	✓	✓	✓	✓
SECURE_RANDOM	✓	✓	✓	✓	✓	✓
	javacard.security.MessageDigest					
SHA	✓	✓	✓	✓	✓	✓
MD5	✓	✓	✓	✓	✓	✓
RIPEMD160	✗	✗	✗	✓	✓	✓
SHA_256	✗	✗	✓	✓	✗	✓
SHA_384	✗	✗	✗	✗	✗	✓
SHA_512	✗	✗	✗	✗	✗	✓
SHA_224	✗	✗	✗	✗	✗	✓
	javacard.security.KeyBuilder					
DES_TRANSIENT_RESET	✓	✓	✓	✓	✓	✓
DES_TRANSIENT_DESELECT	✓	✓	✓	✓	✓	✓
DES	✓	✓	✓	✓	✓	✓
DES3_2KEY	✓	✓	✓	✓	✓	✓
DES3_3KEY	✓	✓	✓	✓	✓	✓
AES_TRANSIENT_RESET	✗	✗	✓	✓	✗	✓
AES_TRANSIENT_DESELECT	✗	✗	✓	✓	✗	✓
AES_128	✗	✗	✓	✓	✗	✓
AES_192	✗	✗	✓	✓	✗	✓
AES_256	✗	✗	✓	✓	✗	✓
RSA_PUBLIC_512	✓	✓	✓	✓	✓	✓
RSA_PUBLIC_736	✓	✓	✓	✓	✓	✓
RSA_PUBLIC_768	✓	✓	✓	✓	✓	✓
RSA_PUBLIC_896	✓	✓	✓	✓	✓	✓
RSA_PUBLIC_1024	✓	✓	✓	✓	✓	✓
RSA_PUBLIC_1280	✓	✓	✓	✓	✓	✓
RSA_PUBLIC_1536	✓	✓	✓	✓	✓	✓
RSA_PUBLIC_1984	✓	✓	✓	✓	✓	✓
RSA_PUBLIC_2048	✓	✓	✓	✓	✓	✓
RSA_PRIVATE_512	✓	✓	✓	✓	✓	✓
RSA_PRIVATE_736	✓	✓	✓	✓	✓	✓
RSA_PRIVATE_768	✓	✓	✓	✓	✓	✓
RSA_PRIVATE_896	✓	✓	✓	✓	✓	✓
RSA_PRIVATE_1024	✓	✓	✓	✓	✓	✓
RSA_PRIVATE_1280	✓	✓	✓	✓	✗	✓
RSA_PRIVATE_1536	✓	✓	✓	✓	✗	✓
RSA_PRIVATE_1984	✓	✓	✓	✓	✗	✓
RSA_PRIVATE_2048	✓	✓	✓	✓	✗	✓

Tab. A.2: Přehled dostupných kryptografických prostředků na kartách - část 2.

Název karty	JCOP 31 v2.2	JCOP 31 v2.3.2	JCOP J3A040	SC Exp. 3.2 144k	SC Exp. 4.X	SC Exp. 6.0
	javacard.security.KeyBuilder					
RSA_CRT_512	✓	✓	✓	✓	✓	✓
RSA_CRT_736	✗	✗	✗	✓	✓	✓
RSA_CRT_768	✓	✓	✓	✓	✓	✓
RSA_CRT_896	✓	✓	✓	✓	✓	✓
RSA_CRT_1024	✓	✓	✓	✓	✓	✓
RSA_CRT_1280	✓	✓	✓	✓	✓	✓
RSA_CRT_1536	✓	✓	✓	✓	✓	✓
RSA_CRT_1984	✓	✓	✓	✓	✓	✓
RSA_CRT_2048	✓	✓	✓	✓	✓	✓
DSA_512	✗	✗	✗	✓	✗	✓
DSA_768	✗	✗	✗	✓	✗	✓
DSA_1024	✗	✗	✗	✓	✗	✓
EC_F2M_113	✓	✓	✗	✗	✗	✗
EC_F2M_131	✓	✓	✗	✗	✗	✗
EC_F2M_163	✓	✓	✗	✗	✗	✗
EC_F2M_193	✓	✓	✗	✗	✗	✗
EC_FP_112	✗	✗	✗	✗	✗	✓
EC_FP_128	✗	✗	✓	✗	✗	✓
EC_FP_160	✗	✗	✓	✗	✗	✓
EC_FP_192	✗	✗	✓	✗	✗	✓
EC_FP_224	✗	✗	✓	✗	✗	✓
EC_FP_256	✗	✗	✓	✗	✗	✓
KOREAN_SEED_TRANSIENT_RESET	✗	✗	✓	✗	✗	✓
KOREAN_SEED_TRANSIENT_DESELECT	✗	✗	✓	✗	✗	✓
KOREAN_SEED_128	✗	✗	✓	✗	✗	✓
	javacard.security.KeyAgreement					
EC_SVDP_DH	✓	✓	✓	✓	✓	✓
EC_SVDP_DHC	✓	✓	✓	✓	✓	✓
EC_SVDP_DH_KDF	✓	✓	✓	✓	✓	✓
EC_SVDP_DH_PLAIN	✓	✓	✓	✓	✓	✓
EC_SVDP_DHC_KDF	✓	✓	✓	✓	✓	✓
EC_SVDP_DHC_PLAIN	✓	✓	✓	✓	✓	✓
	javacard.security.Checksum					
ISO3309_CRC16	✓	✓	✓	✗	✗	✗
ISO3309_CRC32	✗	✗	✗	✗	✗	✓
	javacard.security.KeyPair RSA on-card generation					
RSA RSA_512	✗	✗	✗	✓	✗	✓
RSA RSA_736	✗	✗	✗	✓	✗	✓
RSA RSA_768	✗	✗	✗	✓	✗	✓
RSA RSA_896	✗	✗	✗	✓	✗	✓
RSA RSA_1024	✗	✗	✗	✓	✗	✓
RSA RSA_1280	✗	✗	✗	✓	✗	✓
RSA RSA_1536	✗	✗	✗	✓	✗	✓
RSA RSA_1984	✗	✗	✗	✓	✗	✓
RSA RSA_2048	✗	✗	✗	✓	✗	✓
	javacard.security.KeyPair RSA_CRT on-card generation					
RSA_CRT_512	✓	✓	✓	✓	✓	✓
RSA_CRT_736	✗	✗	✗	✓	✓	✓
RSA_CRT_768	✓	✓	✓	✓	✓	✓
RSA_CRT_896	✓	✓	✓	✓	✓	✓
RSA_CRT_1024	✓	✓	✓	✓	✓	✓
RSA_CRT_1280	✓	✓	✓	✓	✓	✓
RSA_CRT_1536	✓	✓	✓	✓	✓	✓
RSA_CRT_1984	✓	✓	✓	✓	✓	✓
RSA_CRT_2048	✓	✓	✓	✓	✓	✓

Tab. A.3: Přehled dostupných kryptografických prostředků na kartách - část 3.

Název karty	**JCOP 31 v2.2**	**JCOP 31 v2.3.2**	**JCOP J3A040**	**SC Exp. 3.2 144k**	**SC Exp. 4.X**	**SC Exp. 6.0**
	javacard.security.KeyPair DSA on-card generation					
DSA_512	✗	✗	✗	✓	✗	✓
DSA_768	✗	✗	✗	✓	✗	✓
DSA_1024	✗	✗	✗	✓	✗	✓
	javacard.security.KeyPair EC_F2M on-card generation					
EC_F2M_113	✓	✓	✗	✗	✗	✗
EC_F2M_131	✓	✓	✗	✗	✗	✗
	javacard.security.KeyPair EC_FP on-card generation					
EC_FP_112	✗	✗	✗	✗	✗	✗
EC_FP_128	✗	✗	✓	✗	✗	✗
EC_FP_160	✗	✗	✓	✗	✗	✗
EC_FP_192	✗	✗	✓	✗	✗	✗
EC_FP_224	✗	✗	✗	✗	✗	✗
EC_FP_256	✗	✗	✗	✗	✗	✗
	Support for variable public exponent for RSA 1024.					
Allocate RSA 1024 objects	✓	✓	✓	✓	✓	✓
Set random modulus	✓	✓	✓	✓	✓	✓
Set random public exponent	✓	✓	✓	✓	✓	✓
Init. cipher with pub key – random exp.	✓	✓	✓	✓	✓	✓
Use random public exponent	✓	✓	✓	✓	✓	✓

Tab. A.4: Přehled dostupných kryptografických prostředků na kartách - část 4.

B CD SE ZDROJOVÝMI A DALŠÍMI SOUBORY

Přiložené CD obsahuje elektronickou verzi diplomové práce ve formátu PDF, dále ve složce `sources` zdrojové kódy vytvořených programů v programovacím jazyce Java a zdrojové kódy dvou verzí appletů pro platformu Java Card. Veškerý vývoj probíhal ve vývojovém prostředí NetBeans 8.0.2 a všechny přiložené zdrojové soubory jsou připravené jako samostatné projekty pro prostředí NetBeans. Ve složce `application` jsou připravené spustitelné verze vytvořených aplikací včetně knihoven potřebných k jejich spuštění a provozu a také se na CD nachází složka `JavaDOC`, obsahující dokumentaci ke zdrojovým kódům všech přiložených aplikací a appletů. Složka `jcsupport-results` pak obsahuje `.csv` soubory s kompletními výsledky testovacího skriptu JCSupport od Petra Švendy.