VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

POUŽITÍ SMART-KARET V MODERNÍ KRYPTOGRAFII

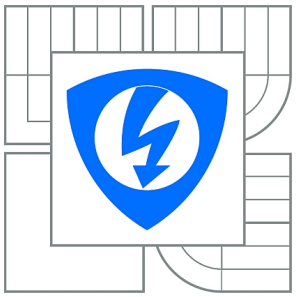
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. MICHAL KOČÍŘ
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

POUŽITÍ SMART-KARET V MODERNÍ KRYPTOGRAFII

THE USE OF SMART-CARDS IN MODERN CRYPTOGRAPHY

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

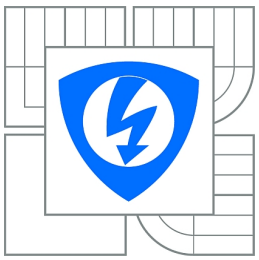
AUTOR PRÁCE — Bc. MICHAL KOČÍŘ

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN HAJNÝ, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Michal Kočíř ***ID:*** 106542
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Použití smart-karet v moderní kryptografii

POKYNY PRO VYPRACOVÁNÍ:

Téma je zaměřeno na využití čipových karet v oblasti bezpečné autentizace a autorizace uživatelů elektronických služeb. Cílem práce je analýza platforem programovatelných čipových karet JavaCard, .NET a MultOS, srovnání jejich parametrů a využitelnosti pro kryptografické protokoly. Výstupem práce je implementace zadaného autentizačního protokolu využívající pokročilou kryptografii na čipové kartě a na platformě PC. Implementace umožní výkonnostní měření jak základních aritmetických primitiv (hash, modulární aritmetické operace, operace s velkými čísly), tak běh pokročilého autentizačního schématu založeného na Schnorrově protokolu.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, William. Cryptography and Network Security: Principles and Practice (5th Edition). USA : Prentice Hall, 2010. 744 s. ISBN 0136097049.

[2] RANKL, Wolfgang a Wolfgang EFFING. Smart Card Handbook. Munich: John Wiley & Sons, 2010. 4. ISBN 978-0-470-74367-6.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

Vedoucí práce: Ing. Jan Hajný, Ph.D.
Konzultanti diplomové práce:

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

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práce obecně pojednává o využití čipových karet MultOS v kryptografických aplikacích. Nejprve jsou popsány dva typy autentizace a to autentizace předmětem se zaměřením na autentizátory a autentizace znalosti. Dále je obsažen popis tzv. anonymní a atributové autentizace.

Dále následuje popis čipových karet se zaměřením na čipové karty MultOS. Zároveň je v této části provedena i analýza programovatelných čipových karet .NET, JavaCard a MultOS.

Vlastní práce je pak zaměřena na praktickou implementaci autentizačního schématu, který je vyvíjen na FEKT VUT v Brně. Komunikace autentizačního protokolu probíhá mezi čipovou kartou MultOS a čtečkou připojenou do PC. Protokol je složen z kryptografických funkcí, jako je generování náhodného čísla, hašovací funkce, modulární umocnění, modulární násobení a rozdílu velkých čísel. Zároveň bylo v praktické části provedeno výkonnostní měření jednotlivých operací.

KLÍČOVÁ SLOVA

Čipové karty, autentizace, kryptografie, revokace, anonymní autentizační systém, soukromí

ABSTRACT

This thesis discusses the general use of smart cards in MULTOS in cryptographic applications. At first is described two types of authentication - the authentication by the subject with focusing on authenticators and the authentication by the knowledge. Furthermore there is the description of the anonymous authentication and attribute authentization.

This is followed by a description of smart cards with a focus on MULTOS cards. There is also performed analysis of programmable smart cards .NET, JavaCard and MULTOS. Practical part is focused on the implementation of an authentication scheme, which is being developed at FEEC. The communication of authentication protocol is between the MULTOS card and reader connected to a PC. The protocol is composed of cryptographic functions such as random number generation, hash function, modular exponentiation, modular multiplication and difference of large numbers. It was also implemented the measurement of specific applications.

KEYWORDS

Smart-cards, authentication, cryptography, revocation, anonymous authentication system, privacy

KOČÍŘ, Michal *Použití smart-karet v moderní kryptografii*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 58 s. Vedoucí práce byl Ing. Jan Hajný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Použití smart-karet v moderní kryptografii“ 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. Janu Hajnému Ph.D. za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

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)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V kapitole 1 bude nejprve rozebrána problematika řízení přístupu a budou popsány funkce jednotlivých části, které řízení přístupu zajišťují. Dále kapitola 1.1 popisuje autentizaci. V této části představím dvě nejčastěji používané autentizační metody: autentizaci předmětem v kapitole 1.1.1 a autentizaci znalostí v kapitole 1.1.2. Třetí autentizační metoda - biometrika se dělí na fyziologické a behaviorální metody. Nicméně tato autentizační metoda není v práci využita, a proto nebude blíže popsána. Více informací o této autentizační metodě lze nalézt v literatuře [3].

Další částí je popis kryptografie v kapitole 2, která slouží k utajení zprávy v takové podobě, která není čitelná například bez znalosti daného klíče. V kapitole 2.1 jsou popsány jednotlivé kryptografické prvky jako je hašovací funkce 2.1.1, digitální podpis 2.1.2, certifikáty a infrastuktura veřejných klíčů 2.1.3. Poté následuje v kapitole 2.2 popis symetrické a v kapitole 2.3 asymetrické kryptografie. Na závěr jsou vyjmenovány kryptografické prvky v anonymní autentizaci v kapitole 2.4, kde je popsána autentizace na bázi nulové znalosti 2.4.1 a Schnorrův identifikační protokol 2.4.2.

Třetí kapitola 3 je věnována čipovým kartám. Postupně popíšu jejich historii, umístění integrovaného obvodu na kartě a komunikaci mezi čipovou kartou a čtečkou. Následuje samostatná část kapitoly 3.1 věnována čipové kartě MultOS, která byla využívána v praktické části diplomové práce. Součástí této kapitoly 3.2 je i analýza platforem programovatelných čipových karet .NET, JavaCard a MultOS.

Vlastní prací je implementace pokročilého autentizačního schématu na čipovou kartu MultOS.

Nejprve tedy v kapitole 4 popisuji programování na samotné čipové kartě MultOS. V této části je popis bezpečnostních vlastností čipových karet MultOS 4.1, tedy postup nahrávání vytvořených aplikací na samotnou kartu MultOS.

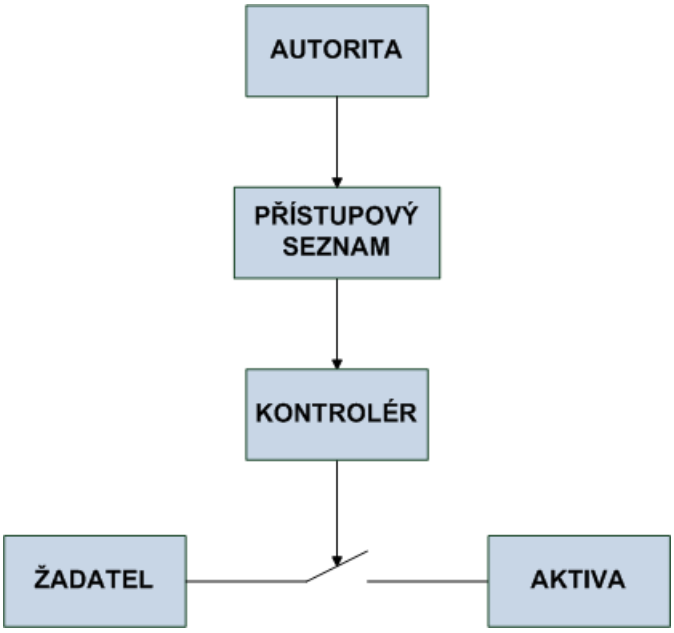
Další kapitola 4.2 je již věnována popisu vytvořených aplikací, které budou následně využity i při vytváření autentizačního protokolu.

Vytvořené aplikace umožňují generování náhodného čísla, hašovací funkce, součet a rozdíl velkých čísel, modulární umocnění a modulární násobení. Zároveň bude provedeno výkonnostní měření jednotlivých aplikací na čipové kartě MultOS.

Poslední kapitola 5 diplomové práce je popis pokročilého autentizačního schématu.

1 ŘÍZENÍ PŘÍSTUPU

Podle [3] je řízení přístupu (Access control) bezpečnostní opatření, jehož smyslem je umožnit uživatelům přístup k aktivům a naopak zase útočníkům v tomto přístupu zabránit. Přístup k aktivům je buď místní, nebo dálkový. Místní je například čtení utajovaných dokumentů v trezoru nebo přístup k počítači či serveru. Dálkový je získání dat ze vzdáleného počítače prostřednictvím sítě. Schéma řízení přístupu je znázorněno na Obr. 1.1.



Obr. 1.1: Řízení přístupu.

Aby žadatel získal přístup k aktivům, musí nejprve předat kontroléru své identifikační údaje (např. uživatelské jméno, které má každý žadatel originální) a provést identifikaci. Následně se provede **autentizace**, kdy kontrolér prověří, zda žadatel je tím, za koho se vydává. Podle přístupového seznamu, kde jsou uloženy identifikační údaje žadatele je ověřeno, zda je uživatel oprávněn (tzv. má přístup k požadovaným aktivům, ke kterým dostal od autority oprávnění).

1.1 Autentizace

Jeden z problémů u řízení přístupu je spolehlivě ověřit identitu žadatele, který požaduje aktiva. Žadatel obvykle udá svoje přihlašovací údaje a prokáže svou identitu. Identitu uživatele lze ověřit několika různými autentizačními metodami:

- autentizace znalostí – tato metoda je založena na tom, že žadatel má určitou znalost, kterou následně prokáže (heslo nebo PIN),
- autentizace předmětem – tato metoda je založena na tom, že žadatel prokáže vlastnictví předmětu tzv. tokenu (např. občanský nebo řidičský průkaz).

Výše uvedené metody mají svá pro a proti. Proto je snahou výše uvedené třídy autentizace kombinovat a využít tzv. *multifaktorovou autentizaci*. Příkladem využití dvou faktorové autentizace je výběr peněz z bankomatu, kdy se musí zadat PIN (znalost) a použít platební karta (předmět).

Nyní budou částečně popsány pouze dvě autentizační metody, a to autentizace na základě předmětu a na základě znalostí, které přímo souvisí s diplomovou prací. Záměrně vynechávám metodu autentizace na základě biometriky, která není v diplomové práci využita. Více informací lze nalézt v literatuře [3].

1.1.1 Autentizace na základě předmětu

Předmět „token“ je v podstatě úložiště, na kterém je uložena důvěrná informace. Tato informace nemusí být snadno zapamatovatelná, takže může být libovolně dlouhá a tedy i odolnější vůči útoku hrubou silou, ale i slovníkovému útoku. Autentizační předměty lze klasifikovat podle typu uložení dat na [3]:

- **Úložiště** je předmět, kam se autentizační informace pouze ukládá. Samotnou autentizaci provádí jiné zařízení - tzv. dokazovací procesor. Úložiště rozdělujeme na *s chráněnými daty* a *s nechráněnými daty*.
    1. Úložiště s chráněnými daty poskytují vyšší úroveň bezpečnosti, protože útočník k získání určitých informací z úložiště musí překonat kryptografickou, nebo přístupovou ochranu. Kryptografickou ochranou je myšleno, že data budou zašifrována, a tedy k jejich získání je zapotřebí znalosti dešifrovacího klíče.
    2. U úložiště s nechráněnými daty lze využít k získání autentizačních dat pouze čtecího zařízení. Jejich využití je u bezkontaktních paměťových karet a u karet s magnetickým proužkem.
- **Autentizátor** je zařízení, které sestává jak z dokazovacího procesoru, tak i z úložiště (např. mikroprocesorová karta). Takové zařízení provádí autentizaci zcela samostatně. Autentizátory jsou sice oproti úložištím dražší, ale poskytují nejvyšší míru bezpečnosti. Jsou to prakticky samostatné počítače, v nichž je dokazovací informace bezpečně uložena a která kompletně zajišťuje autentizaci vůči kontroléru. Zpravidla se jedná o čipové karty, kterým bude věnována samostatná kapitola 3.

1.1.2 Autentizace na základě znalostí

U autentizace znalostí [3] se žadatel prokazuje kontroléru, že zná určitou informaci. Přístupová informace má většinou podobu hesla nebo PINu. Tuto informaci by měl znát pouze uživatel a je tedy důležité ji držet v tajnosti před útočníkem, který by ji mohlo zneužít. Většinou má uživatel přístupovou informaci uloženou ve své paměti a proto by měla být snadno zapamatovatelná. Tento typ autentizace je považován za jeden z nejstarších, který se využívá v informačních systémech. Je to z důvodu, že jsou laciná a jednoduché.

Při autentizaci na základě znalostí se používají tři typy:

- **Login - heslo** je prvním autentizačním mechanismem, kdy se prokazuje přímá znalost. Tento mechanismus je z hlediska bezpečnosti nejslabší, protože pokud útočník zjistí přihlašovací údaje uživatele, může je v budoucnu využívat. Z tohoto důvodu jsou stanoveny určité zásady. Nejjednodušší zásadou je používat bezpečné heslo, které bude obsahovat minimálně 8 znaků a použita bude celá ASCI tabulka. Další zásadou je šifrovat heslo pomocí veřejného klíče a zároveň je neukládat u uživatele.
- **Výzva - odpověď** je druhým, již bezpečnějším autentizačním mechanismem. Zde se již neprokazuje přímá znalost ale pouze se vrací odpověď na výzvu. Nebezpečí zde spočívá v „útoku muže uprostřed“, který může komunikaci odposlechnout a na základě odpovědí zjistit určitou znalost,
- **Důkaz nulové znalosti** je třetím mechanismem, který je založen na prokázání jisté znalosti, aniž by zároveň vyzradil nějakou část chráněné informace. Podrobněji bude tato část probrána v kapitole 2.4.1 s popisem Schnorrova identifikačního protokolu.

2 KRYPTOGRAFIE

Kryptografie je věda, která se zabývá konstrukcí kryptosystémů. Jedním z cílů kryptografie je zahalit zprávu do takové podoby, která není čitelná bez znalosti daného klíče. Technika v kryptografii bývá založena na časové náročnosti hledání řešení, nebo na obtížnosti řešení matematických problémů.

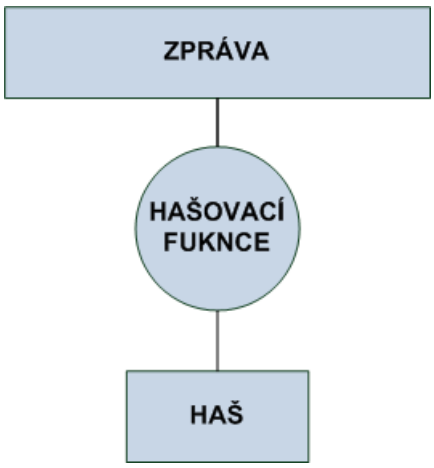
Tato kapitola se zaměřuje na popis jednotlivých funkcí a systémů v kryptografii. Budou zde především popsány obecné kryptografické prvky, dále pak symetrický a asymetrický systém a na závěr je část kapitoly věnována kryptografickým prvkům využitelným v anonymní autentizaci - Schnorrův identifikační protokol.

2.1 Kryptografické prvky

V této kapitole se soustředím na základní kryptografické prvky, které slouží pro ochranu soukromí. Jsou jimi hašovací funkce, digitální podpis a infrastruktura veřejných klíčů.

2.1.1 Hašovací funkce

Hašovací funkce (*hash function*) [3], [20] je matematický algoritmus, který ze vstupní zprávy Z libovolné délky vytvoří krátký řetězec konstantní délky. Hašovací funkce tedy vytvoří tzv. „otisk“, který má vždy stejnou délku bez ohledu na délku vstupní zprávy. Hašovací funkce se formálně zapisuje: $h = H(Z)$. Základní schéma na Obr. 2.1.



Obr. 2.1: Hašovací funkce.

Princip hašovací funkce spočívá v tom, že změna jednoho bitu vstupní zprávy vede k nepředvídatelné změně celého haše. Vlastnost, kterou musí hašovací funkce splňovat, je že ze vstupní zprávy musí vypočítat stále stejný výstup.

Hašovací funkce musí dále splňovat určité požadavky. Prvním požadavek je, aby funkce byla jednocestná, tedy aby neumožňovala sestrojení původní zprávy Z jinou zprávu Z' z daného haše h vypočteného pro zprávu Z. Dalším požadavkem je bezkoliznost, tj. aby nebylo možné v reálném čase nalézt dvě zprávy se stejným hašovacím kódem. Nejběžnější algoritmy jsou *MD5* (Message-Digest algorithm 5) s velikostí výsledného textu 128 b nebo *SHA-1* (Secure Hash Algorith) s velikostí výsledného textu 160 b, z hlediska bezpečnosti je ale spíše preferován algoritmus *SHA-2* s otiskem dlouhým 512 b nebo *Whirlpool*. Více informací lze nalézt v [34].

2.1.2 Digitální podpis

Digitálním podpisem [3] se umožňuje ověření pravosti dokumentů (tzv. autentičnost). Má stejné vlastnosti jako normální podpis, kdy autor má podepisování pod svou kontrolou a zároveň nemůže zpochybnit platnost svého podpisu. Digitálně lze podepsat nejen dokument, ale i libovolný soubor dat nebo přístupová práva. Obvykle se vytváří ve dvou krocích:

Při šifrování digitálního podpisu se nejprve vypočítá haš ze zprávy Z. Tento haš se následně zašifruje soukromým klíčem uživatele a vznikne tak digitální podpis $DS = E(h, SK)$, který je zaslán se zprávou Z příjemci.

Při dešifrování digitálního podpisu nejprve příjemce vypočte haš přijaté zprávy Z'. Následně se digitální podpis dešifruje veřejným klíčem odesílatele $h = D(DS, VK)$ a je-li tento vypočtený obsah totožný s původním obsahem, jsou zpráva a podpis akceptovány.

2.1.3 Certifikáty a PKI - Infrastruktura veřejných klíčů

Certifikáty veřejného klíče, zkráceně „certifikát“ v rámci infrastruktury veřejných klíčů slouží k ověření veřejného klíče. Tento certifikát vlastní pouze určitá osoba a nikoli útočník, který by se za danou osobu mohl vydávat. Certifikáty kromě veřejného klíče mohou dále obsahovat jméno osoby, datum platnosti certifikátu nebo název *CA* (Certifikační Autorita), což je instituce, která využívá všeobecné důvěry a jejich veřejný klíč VK_{CA} je bezpečně předán určité osobě.

Pokud tedy uživatel X chce komunikovat v síti, musí nejpve získat certifikát od certifikační autority. Předloží mu tedy svoje identifikační údaje I_X a veřejný klíč VK_X. CA vytvoří následně certifikát Crt_X, což je digitálně podepsaná zpráva. V této zprávě je sloučena zpráva Z_X a podpis S, který je vypočítán ze soukromého

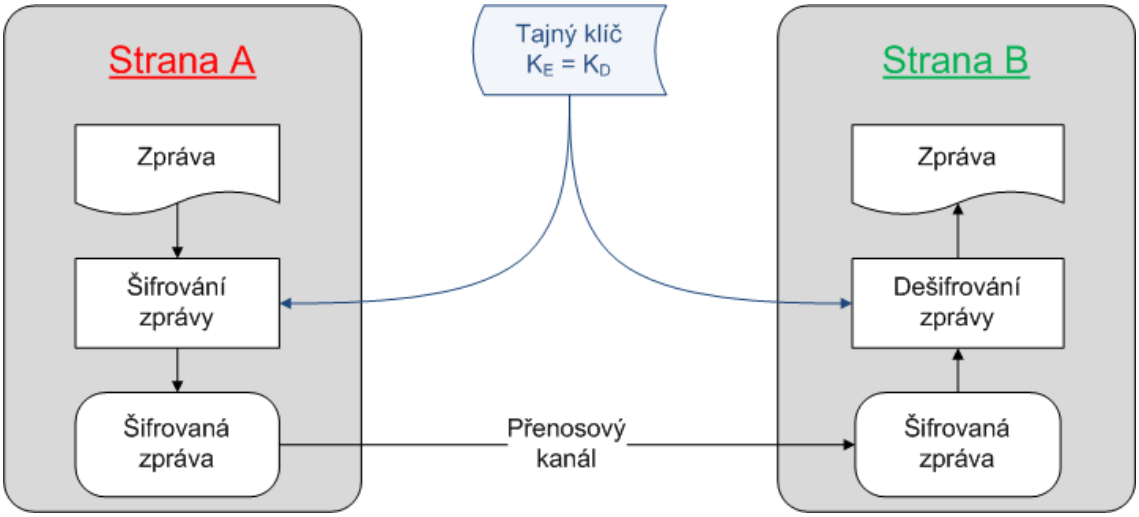
klíče certifikované autority SK_{CA} a zprávy Z_X. Zpráva Z_X je vytvořena sloučením identifikačních údajů I_X a veřejného klíče VK_X. Nyní již uživatel X obdrží digitálně podepsanou zpráv Crt_X a VK_{CA}. Díky klíči a certifikátu autority si může uživatel ověřit, identitu I_X vlastní klíč VK_X. Při komunikaci s jiným uživatelem si předají certifikáty, u kterých si ověří platnost a na základě veřejných klíčů certifikátů mohou bezpečně komunikovat.

$$Crt_X = Z_X||S(Z_X, SK_{CA}), kde Z_X = I_X||VK_X.$$

Více informací o kryptografických prvcích lze nalézt v [3], [10], [20] a [34].

2.2 Symetrická kryptografie

U symetrického kryptosystému [3] (kryptografický systém s tajnými klíči) musí být pro šifrování i dešifrovaní zprávy použit stejný tajný klíč $K_E = K_D$ nebo snadno vypočítatelný $K_E = f(K_D)$. Proto je nutné tento klíč držet v tajnosti a uživatel nesmí dopustit, aby jej odcizila třetí osoba. Výhoda symetrické kryptografie je její relativně vysoká rychlost šifrování.



Obr. 2.2: Symetrická kryptografie.

Symetrické šifrovací algoritmy se dělí na dva typy, proudovou a blokovou:

1. Proudové šifry – jsou oproti blokovým šifrám rychlejší, kdy se každý symbol šifruje samostatně a pracuje se pouze s malými skupinami bitů. Další výhodou proudových šifer je malé šíření chyb, kdy případná chyba ovlivní pouze daný znak. Nevýhody proudových šifer je jejich nízká úroveň difuze, tedy veškerá informace o jednom znaku je transformována opět do jednoho znaku. Další

nevýhodou proudových šifer je náchylnost k úmyslným falzifikacím a modifikacím.

Příkladem proudové šifry je Vernamova šifra. Používá náhodný klíč, který je stejně dlouhý jako otevřený text. Po použití klíče se zničí, což znamená, že není nikdy použit u dvou různých otevřených textů.

2. Blokové šifry – jsou složitější na výpočet, kdy se šifruje skupina symbolů otevřeného textu jako jeden blok. Konstrukce algoritmů symetrických blokových šifer vychází ze dvou základních šifrovacích technik, substituce a transpozice. Velikosti bloku má tedy zásadní význam pro bezpečnost algoritmu, proto při krátkých blocích by bylo možné pro daný klíč vytvořit „slovník“. Převážně jsou bloky předdefinované na velikost bloku 64, 128 nebo 256 bitů. Výhody blokových šifer je vysoká difuze informace otevřeného textu. Další výhoda je imunita vůči narušení, kdy nelze změnit symbol v bloku, aniž by to bylo při dešifrování odhaleno. Nevýhody blokových šifer je zpoždění, kdy je nutné přijmout celý blok. Druhá nevýhoda blokových šifer je šíření chyb, kdy chyba ovlivní všechny ostatní znaků v bloku.

   Snad nejrozšířenějším symetrickým kryptosystémem založeným na blokové šifře je algoritmus *DES* (Data Encryption Standard) používající šifrovací klíč délky 56 bitů. Dnes jej však považujeme za nedostatečný, a proto byl odvozen nový algoritmus 3DES s klíčem dlouhým 112 bitů nebo 168 bitů, tedy v násobcích délky šifrovacího klíče algoritmu DES.

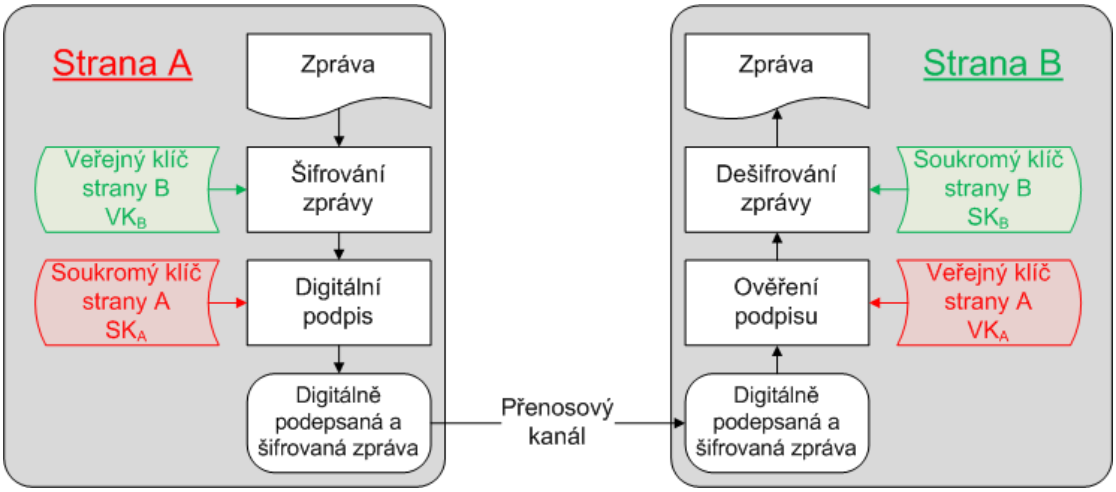
   Další používaný algoritmus je *IDEA* (International Data Encryption Algorithm) nebo *AES* (Advanced Encryption Standard) s délkou klíče 128, 192 nebo 256 bitů. Podrobnější informace lze nalézt v [34].

2.3 Asymetrická kryptografie

Asymetrická kryptografie [3] (kryptografický systém s veřejným klíčem) je metoda, která pracuje s dvěma druhy klíčů. Zpráva se šifruje a následně dešifruje pomocí různých klíčů. To je hlavní rozdíl oproti symetrické kryptografii, která používá k šifrování i dešifrování stejný klíč. Proto u asymetrické kryptografie je možné tajit pouze jeden klíč tzv. soukromý klíč a druhý klíč lze používat veřejně tzv. veřejný klíč. Vlastnost těchto klíčů spočívá v tom, že nelze ze znalosti veřejného odvodit klíč soukromý.

Asymetrickou kryptografii lze rozdělit na dvě části. První způsob znamená, že majitel bude mít veřejný klíč, jímž zprávu zašifruje, a pouze osoba, která bude vlastnit soukromý klíč, může zprávu dešifrovat. Tento systém zajišťuje důvěrnost zprávy. Druhý způsob spočívá v tom, že pouze majitel soukromého klíče může vytvořit kryptogram, který je následně dešifrován veřejným klíčem. Tento princip se využívá

u digitálních podpisů. Problémem digitálního podpisu je, že obsah není žádným způsobem šifrován a každý, kdo má přístup k veřejnému klíči strany A, může číst obsah zprávy. Proto se dva výše zmíněné postupy kombinují, tedy šifrování i podepisování. Na Obr. 2.3 je zobrazena asymetrická kryptografie zajišťující jak autentičnost, tak důvěrnost zprávy.



Obr. 2.3: Asymetrická kryptografie zajišťující autentičnost a důvěrnost.

V praxi se využívá problému faktorizace velkých čísel (rozložení čísla na součin prvočísel), diskrétního logaritmu a sčítání bodů eliptických křivek.

- U **faktorizace velkých čísel** se nejprve vynásobí dvě velká prvočísla. Problém spočívá v tom, že není možné v reálném čase odvodit z výsledku součinu dvě použitá prvočísla. Za bezpečná se v dnešní době považuje čísla o velikosti 2048 bitů. Na tomto problému je založeno schéma RSA.
- Problematika **diskrétního logaritmu** je využita u Diffie-Hellmanova algoritmu, který umožňuje prostřednictvím přenosového kanálu bezpečně domluvit tajný klíč pro symetrický kryptosystém. Nejprve si obě komunikující strany zvolí náhodné velké číslo X, které musí zůstat tajné. Poté vypočítají y z rovnice $y = g^X mod p$. Výpočet y je v tomto případě jednoduchý, ale výpočet neznáme X z čísla y je již náročný a nelze jej v reálném čase provést [3].
- Poslední algoritmus je založen na bázi **eliptických křivek** (Elliptic Curve Cryptography). Opět jsou založeny na řešení diskrétního logaritmu ale jsou obtížněji řešitelné. Na rozdíl od předchozích dvou varianty, tedy faktorizace velkých čísel a diskrétního logaritmu, nemají eliptické křivky exponenciální průběh náročnosti řešení na délce klíče. Proto i při menší délce šifrovacího klíče umožňují větší bezpečnost [29].

Bez ohledu na délku klíčů obecně platí, že asymetrické šifrovací algoritmy jsou výpočetně mnohem náročnější než symetrické algoritmy.

2.4 Kryptografické prvky v anonymní autentizaci

Kryptografické prvky v anonymní autentizaci využívají prvky, které budou níže popsány. V této podkapitole je čerpáno zejména z pramenu [21].

2.4.1 Autentizace na konceptu nulové znalosti

Autentizace s nulovou znalostí je založena na sledu úkolů a odpovědí. Úkolem žadatele tedy je dokázat žadateli, že zná určité tajemství na základě správných reakcí na úkol.

Příkladem protokolu s nulovou znalostí může být ilustrován na ukázce vstupu do jeskyně. V jeskyni jsou dvě cesty a první účastník si zvolí cestu. Druhý účastník následně zvolí úkol, jakou cestou se má účastník A vrátit. Pokud se účastník A vrátil správnou cestou, prokázal tím určitou znalost odpovědi na úkol. Pravděpodobnost úspěchu, že se účastník A vrátí správnou cestou je 0,5. Proto pro zvýšení bezpečnosti se zvýší i počet opakování jednotlivých úkolů a tím pravděpodobnost správného řešení klesá.

2.4.2 Schnorrův identifikační protokol

Jedním z protokolů, který je vytvořen na konceptu nulové znalosti, je Schnorrův identifikační protokol. Jeho bezpečnost je založena na problému diskrétního logaritmu. Protokol byl navržen jako mechanismus tří zpráv neboli Σ-protokol. Je založen na použití prvočísla p v dělitelnosti $p-1$ jiným prvočíslem q. Typicky se používá $p \approx 2^{1024}$ a $q \geq 2^{160}$. Matematický postup lze nalézt v [34].

3 ČIPOVÉ KARTY

V této kapitole je čerpáno zejména z [10], [22], [30] a [33].

Čipové karty 3.1 jsou plastové karty kapesní velikosti s integrovaným obvodem. Slouží k identifikaci a ověření uživatele nebo k ukládání dat. Čipové karty mohou dále poskytnout bezpečnou a spolehlivou autentizaci a autorizaci elektronických služeb. Čipové karty bývají označovány buď jako *ICC* (Integrated Circuit Card) karty, smart karty, nebo také jako tzv. „chytré“ karty.

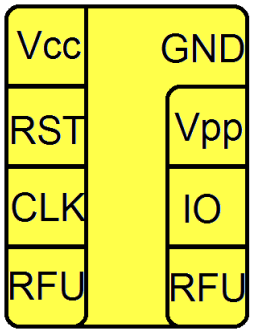
Obr. 3.1: Čipová karta MultOS.

Čipové karty nejprve neobsahovaly zabudovaný mikroprocesor, ale pouze bezpečnostní mechanismy jako je magnetický proužek nebo speciální tisk viditelný pouze pod ultrafialovým zářením.

Dnes se většinou využívají karty dle ISO 7816-1 [1]. Čipová karta je technické zařízení, které poskytuje bezpečnostní funkce spojené s ukládáním soukromých klíčů, tajných klíčů, sdílených tajemství a jiných aktiv držitele čipové karty. Tato kombinace zvýšila bezpečnost a stala se široce využívanou jak v bankovnictví při výběru peněz, tak pro přístup k mobilní síti nebo v dopravě. Nicméně většina čipových karet je určena k využití pouze k jedné aplikaci. Jsou označovány jako jednoaplikační, mají pouze statický přístup k paměti. Nevýhoda je, že jakmile je karta vyrobena, je obtížné změnit její funkčnost. Existují ale i víceaplikační karty, na které je již možné nahrát aplikace, které mohou plnit jak ověření přístupu do firmy tak placení např. obědů. Tato funkce je využívána pomocí logických kanálů, kdy je otevřeno více aplikací v kterémkoliv okamžiku. Nevýhodou je složitost vývoje software.

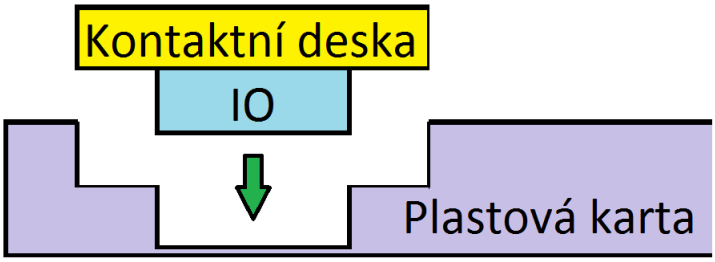
[1] ISO 7816-1 specifikuje fyzikální charakteristiky karty (tepelnou odolnost, ohebnost karty, odolnost proti rentgenovému záření, ultrafialovému záření, elektromagnetickému poli, minimální počet zasunutí karty do čtečky

Základním dělením čipových karet je dělení na kontaktní, bezkontaktní a hybridní. Kontaktní čipové karty mají podle ISO 7816-2 [2] osm kontaktů, jak je znázorněno na Obr. 3.2. Nicméně integrovaný obvod používá pro komunikaci pouze šest kontaktů, z nichž dva kontakty jsou použity pro napájení V_{cc} a V_{pp}. Jeden kontakt je použit pro resetování signálu (RST), aby terminál začal, nebo opakoval komunikaci. Další kontakt slouží jako hodinový signál (CLK) pro regulaci rychlosti provozu. Kontakt (GND) slouží pro uzemnění a poslední využitý kontakt je IO, který slouží pro komunikaci mezi integrovaným obvodem a čtečkou čipových karet. RFU kontakty jsou vyhrazeny pro budoucí použití.



Obr. 3.2: Kontaktní deska.

U kontaktních karet je integrovaný obvod uložen na modulu skládajícím se z kovových vodičů, které jsou následně spojeny s kontaktní deskou a teprve potom vlepeny do vyfrézované dutiny na kartě, jak je znázorněno na Obr. 3.3. V případě bezkontaktních se čip zalévá včetně antény přímo do karty. Z důvodu, že se v diplomové práci nevyužívají bezkontaktní karty, nebudou dále popsány. Více informací lze nalézt v [10].



Obr. 3.3: Umístění integrovaného obvodu na kartě.

[2] ISO 7816-2 specifikuje umístění kontaktů na kartě, jejich rozměr a funkci.

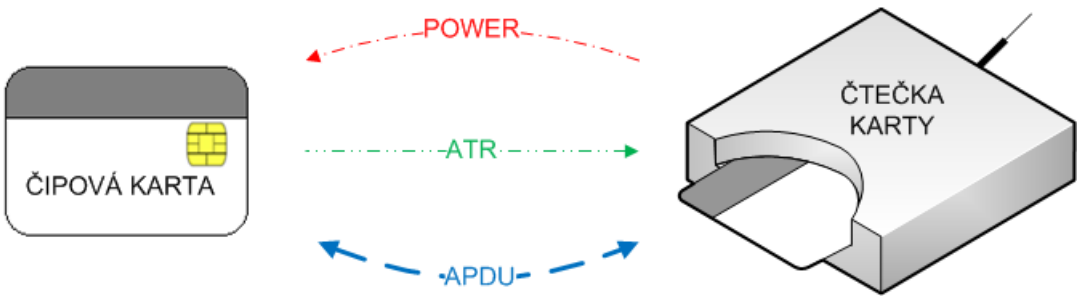
Integrovaný obvod, který je uložený v kartě, je obvykle samostatně vyráběn a je dodáván na pásce. Je to vlastně kompletní mikroprocesorový počítač, který obsahuje ROM paměť, kde je nastaven operační systém od výrobce, dále EEPROM/flash paměť a RAM paměť pro ukládání dat. Integrovaný obvod u kontaktní karty je napájen přes kontaktní desku, která rovněž poskytuje i fyzickou vazbu k sériové komunikaci mezi integrovaným obvodem a terminálem.

Nejčastějšími komunikačními protokoly mezi čtečkou a kartou jsou [10]:

- **T=0** – bajtově orientovaný, kdy nejmenší přenášenou jednotkou je bajt, protokol je asynchronní a polo-duplexní pro sériovou výměnu dat mezi terminálem a kartou.
- **T=1** – jedná se rovněž o asynchronní polo-duplexní sériový protokol mezi čtečkou a kartou, ale tentokrát blokově orientovaný, tj. mezi terminálem a kartou se přenášejí data po celých blocích. Tento protokol zrychluje výslednou komunikaci s kartou, avšak karta musí disponovat větší RAM pro vyrovnávací paměti.

Tyto protokoly řeší pouze fyzickou komunikaci, nicméně z důvodu absence vyrovnávacích pamětí v kartě ovlivňují i logiku odesílání příkazů do karty. Nad těmito protokoly se přenášejí datové pakety, tzv. *APDU* (Application Protocol Data Unit). Pomocí *APDU* se zasílají instrukce kartě, která vrací odpovědi.

Pokud je čipová karta vložena do čtečky, je pomocí kontaktních desek i napájena. Následně čipová karta zašle řetězec bajtů označovaný jako *ATR* (Answer To Reset). *ATR* může být dvojího druhu, buď primární (označeny při upozornění na napájení), nebo sekundární (označeny při resetování, kdy může být zahájena komunikace). Zpráva *ATR* může být velikosti až 33B, ve které jsou obsaženy přenosové parametry, jako je přenosový protokol (T=0 nebo T=1), rychlost datového přenosu nebo sériové číslo karty. ATR jsou nastavovány při výrobě a později již nelze toto nastavení změnit. Po úspěšném odeslání příkazu *ATR* jsou následně zasílány příkazy APDU mezi čipovou kartou a čtečkou 3.4.



Obr. 3.4: Komunikace mezi čipovou kartou a terminálem.

Příkazy *APDU* jsou přenášeny z čtečky karet pomocí *TPDU* (Transport Protocol Data Unit) signálu. *APDU* je zasílaná buď z čtečky na kartu a obráceně. Příkaz *APDU* zasílaný z terminálu na kartu se skládá z povinné a volitelné části, kdy v povinné části je 4B záhlaví skládající se z polí CLA (třída - udává typ příkazu), INS (instrukce - určuje konkrétní příkaz ke spuštění), P1 a P2 (parametry - jsou určeny pro aplikaci). Ve volitelné části jsou umístěna Lc (kóduje délku dat), data (odesílaná data), Le (kóduje očekávaná data, které budou vrácena), jak je znázorněno v Tab. 3.1.

Tab. 3.1: Příkaz APDU – terminál → smart karta.

Záhlaví				**Volitelná část**		
CLA	INS	P1	P2	Lc	DATA	Le

Po zpracování přijatého příkazu čipová karta poskytne odpověď na příkaz *APDU*. Odpověď *APDU* se skládá opět z povinné a volitelné části. Povinná část obsahuje stavové slovo velikosti 2B, které je označeno SW1 a SW2 (výsledek zpracovaného příkazu), a nepovinná část obsahuje data, která mohou být vrácena kartou. Opět je situace znázorněna v Tab. 3.2.

Tab. 3.2: Odpověď APDU - smart karta → terminál.

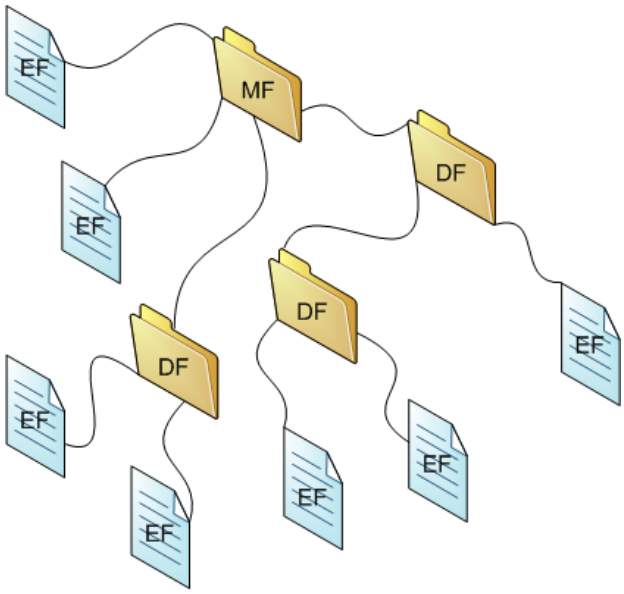
Volitelná část	**Zápatí**	
DATA	SW1	SW2

Jednotlivá stavová slova, která jsou často používána, jsou znázorněna v Tab.3.3

Tab. 3.3: Přehled stavových slov.

Stavová slova	**Význam**
90 00	Úspěšné zpracování
61 XX	Úspěšné zpracování, XX udává velikost dat, která jsou dostupná
6C XX	Chybný parametr Le, XX indikuje správnou velikost
6A 82	Soubor nenalezen

Způsob uspořádání souborů je podle standardu ISO 7816 [3] [22]. Čipové karty obsahují několik různých typů souborů, jak je znázorněno na Obr. 3.5, které budou nyní zkráceně popsány:



Obr. 3.5: Struktura souboru.

Hlavní soubor, který je umístěný na čipové kartě, je tzv. kořenový soubor *MF* (Master File). V tomto souboru jsou umístěny další soubory, které jsou využívány na čipové kartě, jako je *DF* (Dedicated File) a *EF* (Elementary File). *DF* slouží ke spuštění aplikace nebo jako adresář a *EF* slouží k uložení dat, což je definováno podle ISO 7816-4 [4] [22]. Existují čtyři varianty *EF* k uložení dat:

- Transparentní soubor je blok nepřetržitých dat bajtů.
- Soubor s pevnou délkou je rozdělen do několika částí s pevnou délkou.
- Souborem s proměnlivou délkou je soubor, kde každý blok má různou délku.
- Cyklický soubor je podobný souboru s pevnou délkou s tím rozdílem, že rozdíl je při přístupu k souboru.

Z hlediska spouštění aplikací v čipových kartách můžeme též karty dělit na statické s pevně nahranými aplikacemi (většina firemních operačních systémů jednotlivých výrobců karet) a dynamické, do kterých je možné zapisovat nejen data, ale i spustitelný kód. Nejznámější programovatelné čipové karty jsou JavaCard, .NET či MultOS.

[3] ISO 7816 - standard pro kontaktní karty rozšiřující se v lecčems i na bezkontaktní karty.

[4] ISO 7816-4 specifikuje datové příkazy pro komunikaci s kartou, přístupové metody k datům na kartě, zabezpečení komunikace mezi čtečkou a kartou atd.

3.1 Čipová karta MultOS

MultOS je velmi rozsáhlá platforma programovatelných čipových karet, která přináší jednoduchost a inovaci platebních karet a cestovních dokladů. V této kapitole bude popsán pouze architektura MultOS karty, protože vzhledem k tomu, že všechny ostatní vlastnosti jsou již popsány v předchozí kapitole.

3.1.1 Architektura MultOS karty

Čipové karty MultOS jsou jedinečné ve svou bezpečností. Každá aplikace, která bude nahrána na kartu, musí být nejprve certifikována. Na Obr. 3.6 je znázorněna architektura čipové karty MultOS. Nyní popíšu, jak jsou jednotlivé vrstvy architektury poskládány a jak fungují.

Hardware – je fyzická část, která podporuje funkci operačního systému. Funkce jsou psané nativním kódem pomocí virtuálního stroje, který je u všech karet stále stejný bez ohledu na hardware.

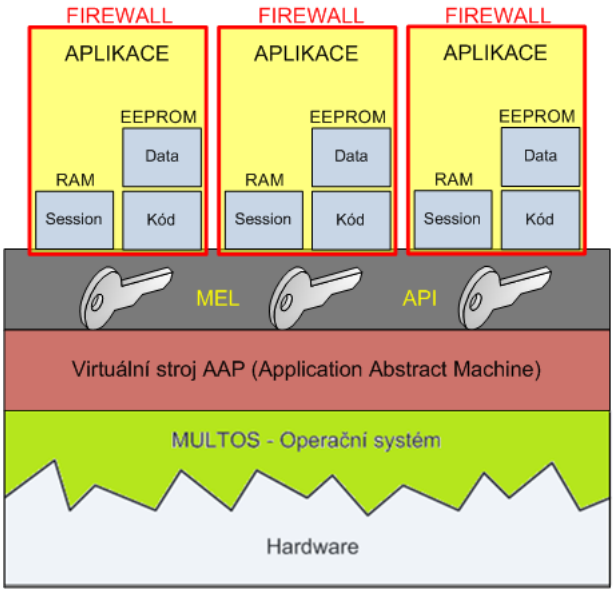
Operační systém MultOS – poskytuje základní komunikaci mezi hardwarem a virtuálním strojem. Rovněž zajišťuje mazání a nahrávání aplikací z karet, dále se stará také o příkazy a odpovědi *APDU*.

Virtuální stroj *AAM* (Application Abstract Machine) – poskytuje standardní instrukce API

MEL API – překládá aplikace psané v assembleru, v jazyku C nebo v Javě. Jak je znázorněno na Obr. 3.6 jsou v této části umístěny i certifikáty (prezentované klíči), které kontrolují, zda jsou nahrané aplikace digitálně podepsané.

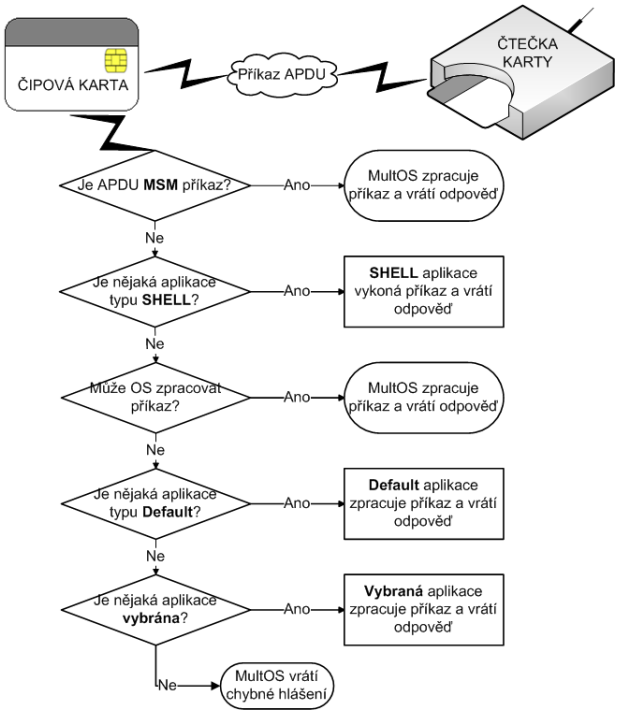
Aplikace a firewall – MultOS využívá aplikační prostor, který je v rámci operačního systému chráněn firewallem. Aplikační prostor se skládá z kódového, datového a relačního prostoru:

- V kódovém prostoru je umístěn spustitelný kód. V této oblasti nelze ani číst ani do ní zapisovat.
- V datovém prostoru jsou uložena data. V této oblasti je možné číst i do ní zapisovat.
- Relační prostor je trvale umístěn v paměti RAM.



Obr. 3.6: Architektura MultOS.

Na Obr. 3.7 je znázorněna komunikace mezi čtečkou a čipovou kartou MultOS pomocí příkazu *APDU*. Aplikace na čipové kartě MultOS využívají tři příkazy *MSM*, *SHELL* a *DEFAULT*. Celý postup je popsaný v literatuře [22].



Obr. 3.7: Komunikace mezi čtečkou a Multos kartou pomocí příkazu APDU.

3.2 Analýza platforem programovatelných čipových karet

V této kapitole budou postupně porovnány programovatelné čipové karty, jako jsou JavaCard [18], .NET karty [26] a MultOS karty [24]. Text bude především zaměřený na srovnání jejich parametrů a využitelnosti v kryptografii. Níže uvedené informace vycházejí z textu [13].

3.2.1 Srovnání parametrů karet

Nejprve budou krátce srovnány parametry jednotlivých karet.

JavaCard

Technologie JavaCard vznikla v roce 1996 a dnes je jedno z nejrozšířenějších multiaplikačních platforem. Je to otevřená platforma, která poskytuje běh aplikací psaných v programovacím jazyku Java Card, což je podmnožina jazyka Java na čipové kartě. JavaCard byla vyvinuta společností Sun Microsystems, což je dceřiná společnost Oracle Corporation. Technologie JavaCard obsahuje i určité bezpečnostní vlastnosti, které převzala ze samotného jazyka Java [18].

Pro srovnání parametrů byly vybrány karty Oberthur Technologies ID-One Cosmo V7.0-A [17] a Gemalto TOP IM GX4 [27]. Parametry jednotlivých karet jsou popsány v Tab. 3.4.

.NET karty

.NET SmartCard [26] byla představena v roce 2002 a obsahují implementaci .NET Framework pro čipovou kartu. Implementace vznikla na žádost firmy Microsoft, protože se stále více rozšiřovala platforma JavaCard. Využívá se programovacího jazyka C#, který je podobný programovacímu jazyku C a Java. .NET SmartCard je velice podobná klasickému Microsoft .NET Framework avšak má určité omezení, které můžou limitovat při vývoji aplikací na čipové kartě. Rozdíly jsou zaznamenány níže [26]:

- datové typy nepracují s plovoucí desetinnou čárkou,
- nepodporuje asynchronní přenos,
- a nepodporuje aplikace pracující s více vlákny.

Pro srovnání parametrů byla vybrána karta Gemalto .NET V2+. Parametry karty jsou poté popsány v Tab. 3.5.

Tab. 3.4: Parametry vybraných Java karet

Specifikace softwaru		
Typ karty	Oberthur ID-One V7.0-A	Gemalto TOP IM GX4
Typ OS	Java kart	Java kart
Přenosový protokol	T=0, T=1	T=0, T=1
Asymetrické algoritmy	RSA do 2048 bitů, EC do 521 bitů	RSA do 2048 bitů
Symetrické algoritmy	DES, 3DES, AES	3DES, AES
Hash funkce	SHA1, SHA2	SHA1
Specifikace hardwaru		
Čip	Atmel AT90SC256144RCFT	S3CC9TC
Výkon	8/16 bit	16 bit
Int./Ext. clock	40 MHz/3,4 MHz	—
RAM paměť	8kB	10kB
ROM/EEPROM paměť	256kB/144kB	384kB/74kB

Tab. 3.5: Parametry .NET karty.

Specifikace softwaru	
Typ karty	.NET V2+
Typ OS	.NET
Asymetrické algoritmy	RSA 2048 bitů
Symetrické algoritmy	3DES, AES
Hash funkce	SHA1,SHA2,MD5
Specifikace hardwaru	
Čip	SLE 88CFX4000P
Výkon	32 bit
Int./Ext. clock	66 MHz/10 MHz
RAM paměť	16kB
ROM/EEPROM paměť	80kB/400kB

MultOS karty

Poslední čipovou kartou, kterou použijeme pro srovnání, je MultOS karta [24]. V porovnání s JavaCard a .NET kartou, MultOS umožňuje vývoj aplikací na obou vyšších úrovních jazyka a jejich sestavení, což poskytuje vývojářům mnohem širší možnosti a lepší přístup k hardwaru. MultOS karty umožňují použít přímo velká čísla pro mo-

dulární operace prostřednictvím aplikačního rozhraní API. Parametry dvou MultOS karet jsou opět zobrazeny v Tab. 3.6.

Tab. 3.6: Parametry MultOS karet.

Specifikace softwaru		
Typ karty	ML2-80K-65	ML3-36K-R1
Typ OS	MultOS	MultOS
Asymetrické algoritmy	RSA 2048, EC 384	RSA 2048, EC 512
Symetrické algoritmy	DES, 3DES, AES	DES, 3DES, AES
Hash funkce	SHA1, SHA2	SHA1, SHA2
Specifikace hardwaru		
Čip	SLE66CLX800PEM	SLE78CLXxxxPM
Výkon	16 bit	16 bit
Int./Ext. clock	30 MHz/7,5 MHz	30 MHz/7,5 MHz
RAM paměť	702+960B	1088+960B
ROM/EEPROM paměť	236kB/78kB	280kB/60kB

3.2.2 Srovnání kryptografických aplikací na kartách

Většina moderních kryptografických systémů jako jsou slepá podpisová schémata, závazkové protokoly, protokoly s nulovou znalostí a skupinové podpisy jsou založeny na jednoduchých kryptografických operacích. Pro většinu těchto systémů je základ použití generování náhodného čísla, hašovací funkce, modulární aritmetické operace nebo operace s velkými čísly. V textu [13] jsou tyto jednotlivé operace popsány a zároveň je zde změřena i doba výpočtu primitivních operací na jednotlivých kartách.

- **Generování náhodného čísla** – měření generování náhodného čísla délky 160 bitů a 560 bitů na jednotlivých kartách.
- **Hašovací funkce** – měření hašovací funkce SHA-1 pro délky 4256, 7328 a 20000 bitů na jednotlivých kartách.
- **Modulární aritmetické operace s velkými čísly** – měření jednotlivých modulárně aritmetických operací jako jsou umocnění, násobení a rozdíl. Pro umocnění je délka modulu 1024 bitů, exponentu, pak délky 160 a 368 bitů. Při násobení jednotlivých operandů je délka použita 1024 a 2048 bitů a při rozdílu je pak délka operandů 400 bitů.

Protokoly nulové znalosti jsou založené na aritmetických operacích v grupě, kde je diskrétní logaritmus těžko spočítatelný. Zejména se používá modulární aritmetika s moduly dlouhými 1024 bitů. Většinou jsou na počítači tyto operace dostupné

v různých knihovnách, ale bohužel na čipových kartách tyto knihovny chybí. Pouze čipová karta MultOS podporuje přímo modulární aritmetiku v *API*(Application Programming Interface).

Výše popsané operace byly podrobeny výkonnostnímu měření a jejich výsledky lze nalézt v literatuře [13].

4 ÚVOD DO PROGRAMOVÁNÍ NA ČIPOVÉ KARTĚ MULTOS

Po teoretické části následuje část práce věnovaná programování na čipové kartě MultOS. Bude zde popsáno, jak je řešena bezpečnost na čipové kartě MultOS. Dále zde bude popsáno vytvoření aplikace pro součet, rozdíl, modulární násobení, modulární umocnění, hašovací funkce SHA1 a generování náhodného čísla.

4.1 Bezpečnost na čipových kartách MultOS

V dnešní době existují různé způsoby komunikací, a proto je potřeba zaručit jejich bezpečnost. Jednou z možností komunikace je využití čipových karet. Proto byl vytvořen bezpečný operační systém MultOS, který umožňuje běh aplikací v bezpečném prostředí. Tento operační systém může být implementován do různých čipových karet. V praktické části byla použita kontaktní čipová karta MultOS ML2-80K-65. Bezpečnost na čipových kartách MultOS je zaručena tím, že před nahraní aplikace na kartu musí být aplikace nejprve certifikovány. Zkrácený popis nahrání jednotlivých aplikací na kartu je představena níže:

- Nejprve je tedy nutné napsat samotný zdrojový kód. Byl využit obyčejný textový editor **Notepad2**, který je volně dostupný na Internetu. Vytvořená aplikace byla napsána v programovací jazyku C. Dokumentace k psaní aplikací na čipovou kartu MultOS je dostupná ze stránek MultOS.
  Každá vytvořená aplikace by měla obsahovat minimálně tyto části:
  1. **Atributy**: umožňují specifikaci určitých parametrů aplikace. Ve zdrojovém kódu se většinou píšou na začátku. Existují ale různé varianty. V našem případě byla použita struktura s nastaveným hexa slovem.

     ```
     #pragma attribute("aid","DE AD BE EF")
     ```

     Další atributy, které je nutno použít, slouží pro upřesnění, v jaké části paměti se nacházejí používané datové struktury.

     ```
     #pragma melpublic
     #pragma melstatic
     ```

     Statická paměť slouží pro ukládání informací, které jsou dlouhodobě uchovávány. Do veřejné paměti se definuje struktura, která se využívá při komunikaci mezi kartou a terminálem.
  2. **Konstanty**: slouží pro lepší přehlednost ve zdrojovém kódu.

     ```
     #define TEST_CLA                0xAA
     #define INS_GET_ADD             0x10
     ```

```
#define INS_GET_SUBTRACT          0x11
#define INS_GET_MULTIPLICATION    0x12
#define INS_GET_EXPONENTIATION    0x13
#define INS_GET_SHA1              0x14
#define INS_GET_RANDOM            0x15
#define INS_GET_AUTHENTICATE      0x16
```

Další konstanty a jejich vysvětlení lze nalézt v příloze této práce.

3. **Knihovny**: jsou dostupné s prostředím SmartDeck na stránkách MultOS. V našem případě byly použity knihovny.

```
#include <multosarith.h>
#include <multoscrypto.h>
#include <string.h>
#include <multoscomms.h>
#include <RSA.h>
```

4. **Deklarace proměnných**: je možné použít union nebo structure. SmartDeck podporuje následující datové typy:

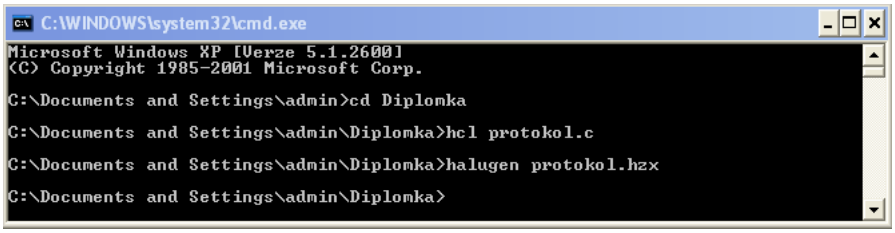
Datové typy	**Definice**	**Velikost**
BOOL	boolean	1 bajt
BYTE	unsigned byte	1 bajt
SBYTE	signed byte	1 bajt
WORD	unsigned word	2 bajty
SWORD	signed word	2 bajty
DWORD	unsigned double word	4 bajty
SDWORD	signed double word	4 bajty

5. **CheckCase**: při komunikaci s IFD dostává čipová karta určitá data. Je však důležité, aby věděla, jakou strukturu mají, zda např. IDF má očekávat odpověď. Když je aplikací zaslán příkaz APDU, vidí příjemce pouze hlavičku. Pomocí funkce *CheckCase* s udaným ISO Case může MUTLOS interpretovat přijatá data a aplikace, s nimiž pak může pracovat. Volání této funkce pro data pouze na výstupu bude ISO Case vypadat následovně:

```
if (!CheckCase(NODATAIN_DATAOUT))
 ExitSW(ERR_ISOCASE);
```

- Nyní pokud tedy máme již napsaný celý zdrojový kód, musí se provést určité operace, které nám umožní aplikaci nahrát na čipovou kartu. Zdrojový kód je dostupný v příloze diplomové práce. Nejprve tedy využijeme programu

MULTOS SmartDeck Debugger, který slouží k překladu vytvořené aplikace. Pracuje se s ním pomocí příkazového řádku, kde se nejprve zavolá příkaz *hcl test.c*, pomocí kterého je vytvořen soubor **test.hzx**. Tento soubor znovu v příkazovém řádku zavoláme pomocí příkazu *halugen test.hzx*, při kterém je vytvořen soubor **test.alu**, se kterým budeme dále pracovat. Příkazový řádek je zobrazen na Obr. 4.1.

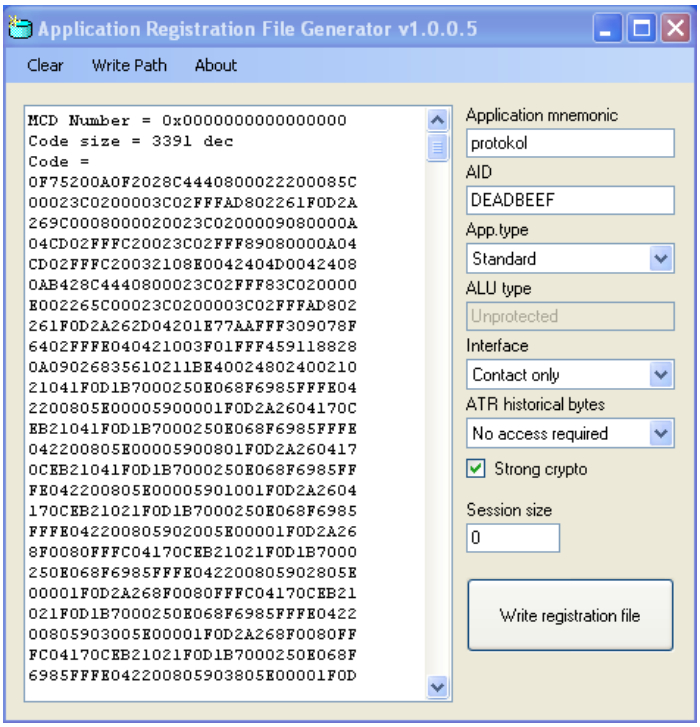


Obr. 4.1: Ukázka jednotlivých příkazů.

- Dále se využit program **Application Registration File Generator v1.0.0.5**. Jak bylo v teoretické části vysvětleno, aby bylo možné na karty MultOS nahrávat vlastní aplikace, je zapotřebí použit certifikát. Před vytvořením certifikátu je ale nutné vytvořit soubor **test.aif**. K tomu slouží program Application Registration File Generator, do kterého bude vložený již vytvořený soubor **test.alu**. Jednotlivá nastavení jsou zobrazena na Obr. 4.2.
- Nyní je všechno předpřipravené k tomu, aby byly vygenerovány samotné certifikáty na stránkách StepXpressu. Po přihlášení na stránky StepXpressu se v záložce Applications vytvoří nová aplikace, kam je zadána cesta vytvořeného souboru **test.aif**.
  Vytvořené certifikáty Obr. 4.3 budou následně uloženy do PC, kde budou později využity v programu Mutil.
- A na závěr je využit program **Mutil**. Po vložení karty MultOS do čtečky je nejprve ověřeno, zda karta s čtečkou správně komunikuje. Kontrolu provedeme v záložce Exchange APDU. Nahraní aplikace na kartu je provedeno v záložce Load Live. Zde musíme vložit jak vytvořený soubor **protokol.alu**, tak vygenerovaný certifikát. Poté již se stiskneme tlačítko Load a provedeme nahrání aplikace na kartu, jak je znázorněno na Obr. 4.4.

4.2 Vytvořené jednotlivé aplikace

Hlavní částí praktické části bylo vytvoření pokročilého autentizačního schématu. Toto schéma se skládá z různých operací jako je rozdíl, modulární násobení, modulární umocnění, hašovací funkce SHA1 a generování náhodného čísla. Dříve než

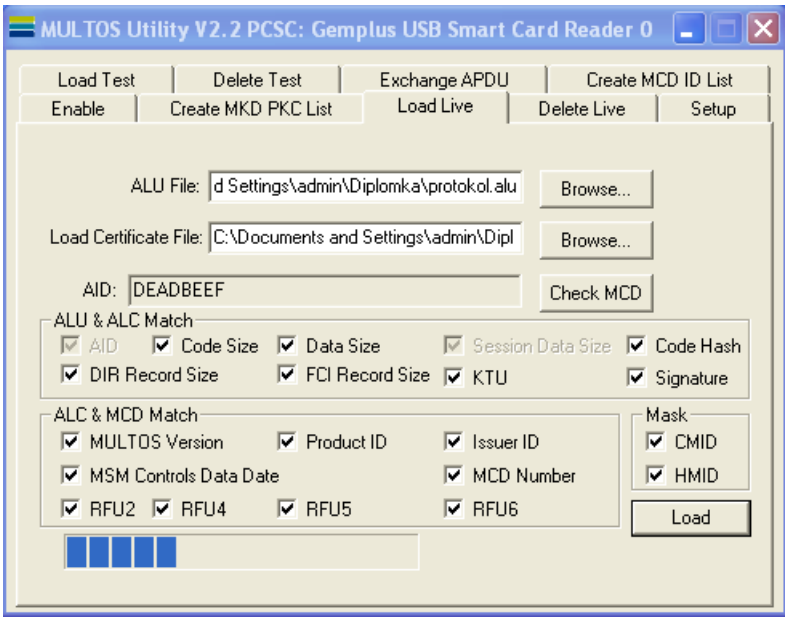


Obr. 4.2: Nastavení generátoru.

Obr. 4.3: Vytvořené certifikáty.

přistoupíme k samotnému popisu autentizačního schématu, věnuji část této kapitoly jednotlivým operacím. Budou zde nejen zkráceně popsány jednotlivé operace, ale také znázorněno i jejich výkonnostní měření. Z tohoto důvodu bylo zároveň vytvořeno grafické rozhraní v programovacím jazyku Java, kde je provedenou samotné měření. Jak již bylo zmíněno, bude využita čipová karta MultOS. Díky tomu, že operace jsou psané v programovacím jazyku C, je možné využít dokumentaci *API* (Application Programming Interface), která je volně dostupná na stránkách MultOS. Při měření je každá operace 25 krát zopakována a tato hodnota je zobrazena v Tab. 4.1.

Generování náhodného čísla – parametr je 8 bajtové pole kam bude zapsané náhodné číslo.



Obr. 4.4: Nastavení Load Live.

```
void GetRandomNumber (BYTE result[8])
```

Tato funkce generuje osmi bajtové náhodné číslo a zapíše se do hodnoty result[8]. Výkonnostní měření generování náhodného čísla bylo provedeno pro délky 640b.

```
case INS_GET_RANDOM:
 while (i < MODLENA)
 {
  GetRandomNumber(num);
  memcpy(apdu_data.ndata.N + i, num, 8);
  i = i + 8;
 }
```

Součet dvou čísel – použité parametry při součtu dvou čísel:

- const BYTE blockLength: velikost bloků pro součet
- const BYTE *result: adresa bloku, kde bude zapsaný výsledek operace součtu
- BYTE *block1: adresa prvního bloku
- BYTE *block2: adresa druhého bloku

```
void BlockAdd (const BYTE blockLength, const BYTE *result,
 BYTE *block1, BYTE *block2)
```

Při součtu dvou čísel se hodnoty uložené v block1 a v bloku2 sečtou a výsledek se vypíše do result. Velikosti jednotlivých částí závisí na blockLength.

V praktické části diplomové práce byly názvy jednotlivých operandů jak je znázorněno níže. Pro výkonnostní měření součtu budou jednotlivé operandy délky 400b.

```
case INS_GET_ADD:
 ADDN(MODLEN, num_a, num_b, num_c);
```

Rozdíl dvou čísel – použité parametry při rozdílí dvou čísel:

- const BYTE blockLength: velikost bloků pro rozdíl
- const BYTE *result: adresa bloku, kde bude zároveň zapsaný výsledek operace
- BYTE *block1: adresa prvního bloku
- BYTE *block2: adresa druhého bloku

```
void BlockSubtract (const BYTE blockLength, const BYTE *result,
 BYTE *block1, BYTE *block2)
```

Při rozdílu dvou čísel se hodnoty uložené v block1 a v bloku2 odečtou a výsledek se vypíše do result. V praktické části diplomové práce byly názvy jednotlivých operandů jak je znázorněno níže. Pro výkonnostní měření rozdílu budou jednotlivé operandy délky 400b.

```
case INS_GET_SUBTRACT:
 SUBN(MODLEN, num_a, num_b, num_c);
```

Modulární násobení – použité parametry při modulárním násobení:

- WORD modulusLength: délka modula
- BYTE *block1: adresa prvního operandu
- BYTE *block2: adresa druhého operandu
- BYTE *modulus: adresa modula

```
void ModularMultiplication (WORD modulusLength, BYTE *block1,
 BYTE *block2, BYTE *modulus)
```

Při modulárním násobení se výsledek vypíše do block1. V praktické části diplomové práce byly názvy jednotlivých operandů jak je znázorněno níže. Pro výkonnostní měření modulárního násobení budou všechny operandy nastaveny na stejnou délku 512b a 1024b.

```
case INS_GET_MULTIPLICATION:
 ModularMultiplication(MODLEN, num_c, num_a, num_b);
```

Modulární umocňování – použité parametry při modulárním umocňování:

- WORD exponentLength: délka exponentu
- WORD modulusLength: délka modula
- BYTE *exponent: adresa exponentu

- BYTE *modulus: adresa modula
- BYTE *input: adresa vstupní hodnoty
- BYTE *output: adresa výstupu, kde bude zobrazen výsledek operace

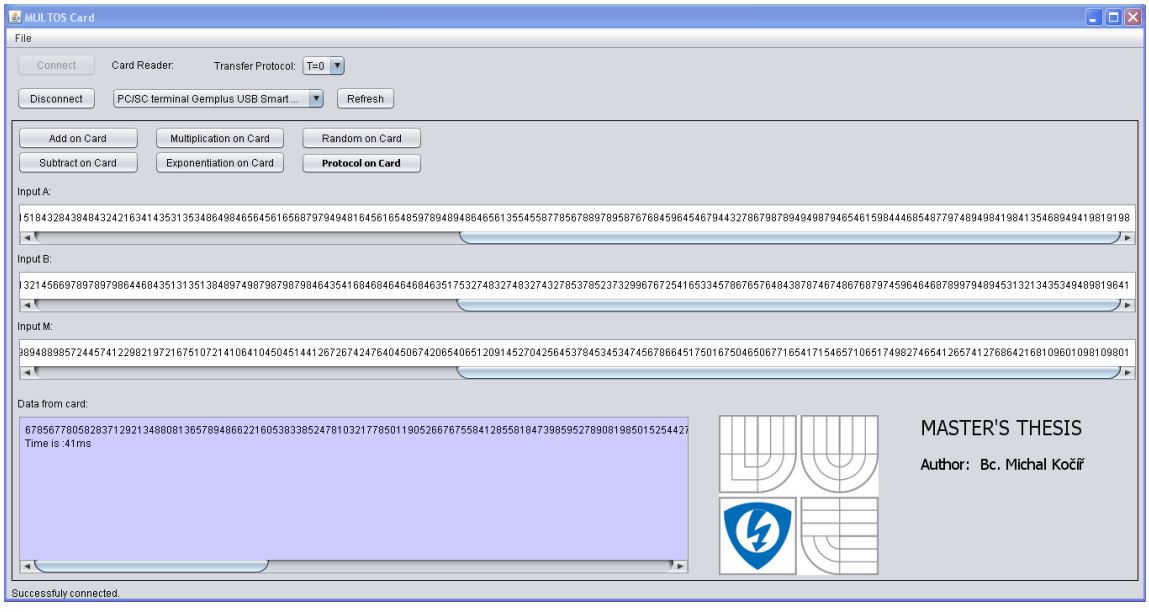
```
void ModularExponentiation (WORD exponentLength,WORD modulusLength,
BYTE *exponent, BYTE *modulus, BYTE *input, BYTE *output)
```

Při modulárním umocňování se výsledek vypíše do output. V praktické části diplomové práce byly názvy jednotlivých operandů jak je znázorněno níže. Pro výkonnostní měření modulárního umocňování bude délka modula 1024b. Exponenty potom budou mít délku 160b a 368b.

```
case INS_GET_EXPONENTIATION:
 ModularExponentiation(MODLEN, MODLEN, num_b, num_c, num_a, num_d);
```

4.3 Měření výkonnosti vytvořených operací

Dále bylo vytvořené grafické rozhraní v programovacím jazyku Java. Jak je zobrazeno na Obr. 4.5, grafické rozhraní se skládá ze dvou částí. Vrchní část slouží pro připojení čipové karty. Spodní část poté slouží k použití jednotlivých operací.



Obr. 4.5: Zobrazené grafické rozhraní.

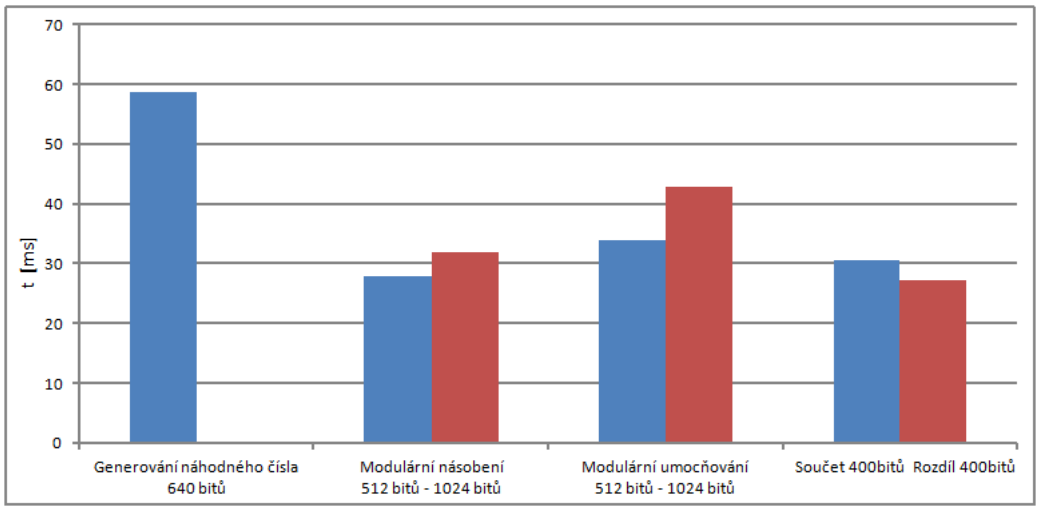
Grafické rozhraní zároveň umožňuje zobrazení výkonnostní měření na čipové kartě MultOS, kdy je tedy umožněno měření šesti aplikaci jako je součet, rozdíl, modulární násobení, modulární umocňování, generování náhodného čísla a hlavně

výkonové měření vytvořeného autentizačního schématu, které bude blíže popsané v kapitole 5.

Tab. 4.1: Výkonnostní měření vytvořených operací na čipové kartě MultOS.

Aplikace	Velikost [bit]	Změřené hodnoty [ms]					Průměr [ms]
Generování náhodné čísla	640	66	56	57	54	62	59
Modulární násobení	512	32	27	25	26	29	28
	1024	37	29	32	34	28	32
Modulární umocňování (exponent 1024b)	512	38	36	30	36	29	34
	1024	45	42	47	42	39	43
Součet	400	34	30	29	33	27	31
Rozdíl	400	32	27	25	23	29	27

Výsledky jednotlivých měření na čipové kartě MultOS jsou uvedeny v Tab. 4.1. Uvedené časy jsou pouze orientační, protože se v nich promítá i doba přenosu mezi čipovou kartou a čtečkou. Průměrné hodnoty jsou poté zobrazeny v Obr. 4.6.



Obr. 4.6: Výkonnostní měření vytvořených operací na čipové kartě MultOS.

5 IMPLEMENTACE AUTENTIZAČNÍHO SCHÉMATU

V diplomové práci je popsané autentizační schéma, které je vyvíjeno na fakultě FEKT VUT v Brně. Vyvíjené autentizační schéma se nazývá „Unlinkable Attribute-Based Credentials with Practical Revocation on Smart-Cards“, které je blíže popsané v [15]. Autentizační schéma umožňuje revokaci odvolaných uživatelů a deanonymizaci útočníků na komerčně dostupných čipových kartách. Praktickou částí diplomové práce je tedy implementace tohoto schématu do čipové karty MultOS a její výkonnostní měření.

V této kapitole je čerpáno z pramenu [15].

5.1 Koncepce schématu

Atributové pověření bylo navrženo k zlepšení ochrany soukromí uživatele. Při použití atributového pověření může uživatel anonymně prokázat některý z atributů. Atribut může být prezentován osobními údaji uživatele, jako například věk, občanství nebo řidičský průkaz. Na rozdíl od klasického ověření, identita uživatele nejsou nikdy vydány. To znamená, že ověřovací proces je anonymní a tím se zvětšuje ochrana soukromí uživatele. Při použití atributového pověření založeného na čipových kartách by uživatelé mohli prokázat svoje atributy, aniž by prozradili svoji totožnost, případně jiné soukromé informace, které by mohl ověřovatel zneužít. S rostoucím počtem elektronických služeb a při blížícím se využívání elektronických občanských průkazů, je nutné zvýšit ochranu a soukromí uživatele. Atributové pověření by tedy mělo být schopno poskytnout následující funkce:

- **anonymita**: identita uživatele by měla zůstat při ověřování ověřovatelem stále skryta
- **nesledovatelnost**: vydavatel není schopen dohledat vydané atributy a jejich vlastníky
- **nespojitelnost**: ověření relace jednoho uživatele jsou vzájemně nespojitelné
- **selektivní zveřejnění atributů**: uživatel může zpřístupnit ověřovateli pouze určité soukromé atributy pro ověření
- **nepřenosnost**: půjčování pověření je zabráněno
- **revokace**: zrušení neplatné, ztracené, odcizené karty nebo při vypršení její platnosti
- **identifikace nečestného a podvádějícího uživatele**: ačkoli je prokázání atributů anonymní, musí existovat způsob odhalení nečestného a podvádějící uživatele

Zejména je obtížná revokace a identifikace nečestného a podvádějící uživatele pomocí stávajících systémů, a proto byl vytvořen nový systém, který všechny popsané funkce umožňuje.

5.2 Současný stav v revokačních technikách

Pro zajištění soukromí uživatele při ověření relace (např. při použití karty při vstupu do budovy), by měl uživatel zůstat zcela anonymní a nemělo by tedy docházet k odhalení jeho osobních informací, které by mohly být později dohledány. Avšak v případě, kdy je karta uživatele odcizena, ztracena nebo zničena je nutné po ověření uživatele kartu zrušit, aby nedocházelo v budoucnosti k jejímu zneužití. Zároveň ale musí existovat i způsob, kdy při porušení určitých pravidel společnosti bude identita uživatele odhalena. Současné revokační techniky toto neumožňují. Více informací lze nalézt v [15].

Proto byl vytvořen nový autentizační protokol nespojitelného atributového pověření s praktickou revokací. Tento protokol má následující vylepšení:

- Okamžité zrušení platnosti pověření, což znamená, že nemusí čekat na vypršení určité doby platnosti.
- Revokace je dostupná, jak pro vydavatele, tak pro ověřovatele. Oba dva tedy mohou zahájit revokaci uživatele.
- Platný uživatel nemůže převzít nebo stáhnout hodnoty od zrušeného uživatele.
- Ověřovací relace běží pouze mezi uživatelem a ověřovatele a není potřeba komunikovat s jinou stranou.

Revokace ověření je velmi důležitý prvek, ale v některých případech nestačí pouze odstranit uživatele ze systému, ale v případě vzniku škody potřebuje poskytovatel služeb určitou techniku, která mu umožní, aby se útočník zodpovídal za způsobené škody. Proto schéma umožňuje i zrušení určitých funkcí, jako je nespojitelnost a anonymita. Pokud je potřeba zkontrolovat minulost uživatele, využije se funkce zrušení nespojitelnosti. Jakmile ale uživatel porušil závažné pravidla, je možné zrušit jeho anonymitu, aby byl zodpovědný za své činy.

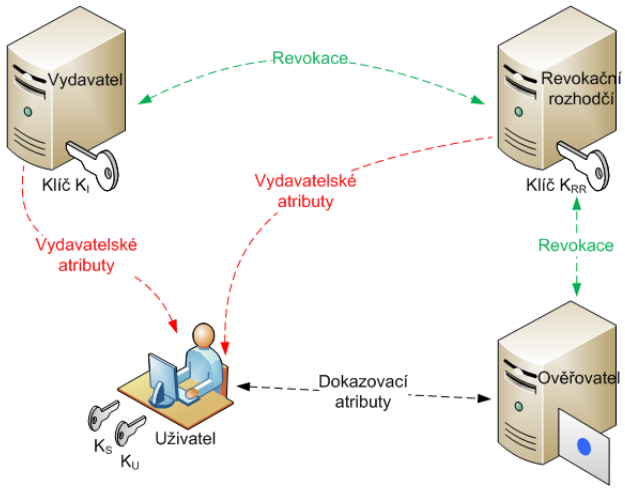
Vzhledem k těmto skutečnostem je nutné revokační techniku silně chránit před zneužitím. Proto se schopnost revokovat rozšířila na více entit. V tomto systému musí spolupracovat ověřovatel, vydavatel a revokační rozhodčí na revokaci soukromých údajů uživatele. Tímto způsobem se omezí možné zneužití jedním ze subjektů (**V** + **RR**). Kromě toho, má uživatel svobodnou volbu, kdy si může zvolit atribut od vydavatele, kterému nejvíce důvěřuje.

5.3 Schéma navrženého atributového pověření

V této části je popsáno schéma pro atributové ověření. Zde jsou představeny jednotlivé subjekty a využívané kryptografické funkce [15].

5.3.1 Fáze schématu

Navrhované schéma se skládá ze čtyř částí. Některé mají v držení tajné klíče, jak je znázorněno na Obr. 5.1.



Obr. 5.1: Schéma systému AASR.

- **Vydavatel – V**: subjekt, který vydává osobní atributy pro uživatele. Vydavatel, revokační rozhodčí a ověřovatel spolupracují při zrušení anonymity a odhalení uživatele se zlými úmysly. Vydavatel vlastní klíč K_V.
- **Revokační rozhodčí – RR**: subjekt, který generuje systémové parametry *params*, spolupracuje s vydavatelem při vydávání atributů uživatele, a jak již bylo popsáno, spolupracuje i při zrušení anonymity s vydavatelem a ověřovatelem. Revokační rozhodčí funguje jako záruka ochrany osobních údajů, protože rozhoduje o typu revokace (revokace pověření, revokace nespojitelnosti nebo revokace anonymity) na základě důkazů poskytnutých ověřovatelem. Revokační rozhodčí není sám schopný revokovat nebo odhalit soukromé informace uživatele, ale pro zvýšení bezpečnosti pouze ve spolupráci s ověřovatelem a vydavatelem. Revokační rozhodčí vlastní klíč K_{RR}.
- **Uživatel – U**: subjekt, který je držitelem čipové karty s vydanými atributy. Uživatel může anonymně prokázat vlastnictví atributů pomocí čipové karty.

Každá čipová karta má tajný jedinečný klíč K_U, určený pro generování důkazních atributů. Navíc uživatel generuje i tajný klíč K_S pro ověření, které je zároveň zcela náhodné, aby bylo nespojitelné s přecházející operací.

- **Ověřovatel – O**: subjekt, který ověřuje vlastnictví uživatelských atributů. Používá databázi k ověření relace a evidence porušených pravidel. Ověřovatel může požádat revokačního rozhodčího o revokaci uživatele. Pokud revokační rozhodčí rozhodne, že je revokace správná, uživatelský klíč K_U bude revokován a tím může dojít i k zveřejnění identity podvádějícího uživatele.

5.3.2 Fáze atributového pověření s praktickou revokací

Atributové pověření se skládá ze čtyř fází. V diplomové práci bude popsána pouze jedna fáze atributového pověření, a to protokol běžící mezi uživatelem a ověřovatelem. Je to z důvodu, že praktická část diplomové práce byla zaměřena na vytvoření tohoto protokolu mezi uživatelem a ověřovatel. Ostatní fáze atributového pověření jsou podrobněji popsány v literatuře [15]

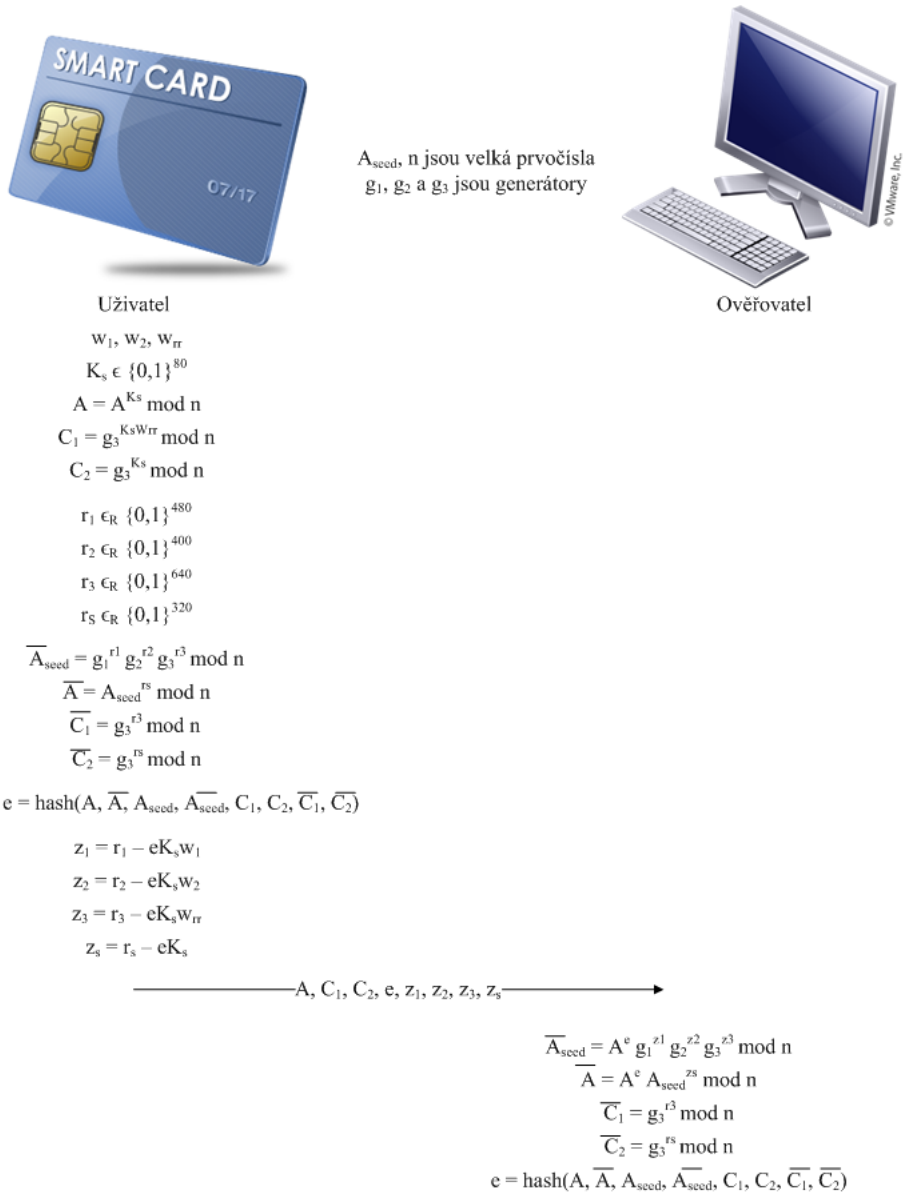
$proof \leftarrow$ **dokazovací atributy** $(params, K_U)$: tento protokol běží mezi uživatelem a ověřovatelem, jak je i znázorněno na Obr. 5.1. Uživatel získá z předcházející fáze na svoji čipovou kartu atribut, což je vlastnost uživatele. Tuto vlastnost uživateli vydává revokační rozhodčí a vydavatel.

Na Obr. 5.2 je pak znázorněn celý autentizační protokol, který je implementován jak do čipové karty Multos, tak do PC.

Uživatel má uložené tajné klíče w_1, w_2 a w_{RR}, kterými se následně prokazuje ověřovateli. Prokazuje to ale takovým způsobem, že celou operaci může provést vícekrát za sebou. Ověřovatel tak nebude schopný vědět, zda se jedná stále o stejného uživatele nebo již jiného. To je způsobeno využitím čísel K_S, r_1, r_2, r_3 a r_{SS}, které jsou náhodně generovány. Každá relace bude zcela náhodná a nespojitelná a nesmí se nikdy objevit dvě stejná čísla. Toto lze zkontrolovat v praktické části, například pozorováním čísel z_1, z_2, z_3 a z_S, které jsou stále náhodné. Proto, když se uživatel prokazuje ověřovateli v různých časových okamžicích, nebude ověřovatel schopný přesně určit, zda se jedná o stejného uživatele nebo již jiného. Jediné, co ale ověřovatel ví, že žádost pochází od uživatele, který zná tajné klíče w_1, w_2 a w_{RR}.

5.4 Implementace autentizačního schématu

Vytvořený program představuje implementaci komunikace mezi uživatelem a ověřovatelem tedy popis části $proof \leftarrow$ dokazovací atributy $(params, K_U)$. Tato část je podrobněji popsaná v 5.3.2. Autentizační protokol byl úspěšně vyzkoušen na operačním systému Microsoft Windows XP 32bitů.



Obr. 5.2: Autentizaceční protokol mezi uživatel – ověřovatel.

Pokročilé autentizační schéma se skládá z různých operací, které jsou popsány v kapitole 4.2. Nicméně samotná implementace autentizačního schématu do čipové karty MultOS je poněkud složitější, kdy se využívá kombinace jednotlivých operací. Například parametr $\bar{A}_{seed} = g_1^{r_1} g_2^{r_2} g_3^{r_3} \bmod n$ je vytvořen kombinací modulárního umocňování a modulárního násobení. Zároveň se využilo pomocných parametrů POM a POM1.

Výpočet parametru $\bar{A}_{seed} = g_1^{r_1} g_2^{r_2} g_3^{r_3} \bmod n$:

```
ModularExponentiation(MODLEN,MODLEN,R1,N,G1,POM);
ModularExponentiation(MODLEN,MODLEN,R2,N,G2,POM1);
```

```
ModularMultiplication(MODLEN,POM,POM1,N);
ModularExponentiation(MODLEN,MODLEN,R3,N,G3,AS1);
ModularMultiplication(MODLEN,AS1,POM,N);
```

Dalším příkladem využití kombinace jednotlivých operací jsou parametry z_1, z_2, z_3 a z_S, které jsou vytvořeny kombinací modulárního násobení a rozdílu velkých čísel. Zároveň se při této operaci musely jednotlivé parametry upravit na stejnou velikost, kdy např. rovnice $z_1 = r_1 - eK_S w_1$ byla upravena na velikost 480 bitů a z_2 na velikost 400 bitů. Celý zdrojový kód je napsán v příloze diplomové práce.

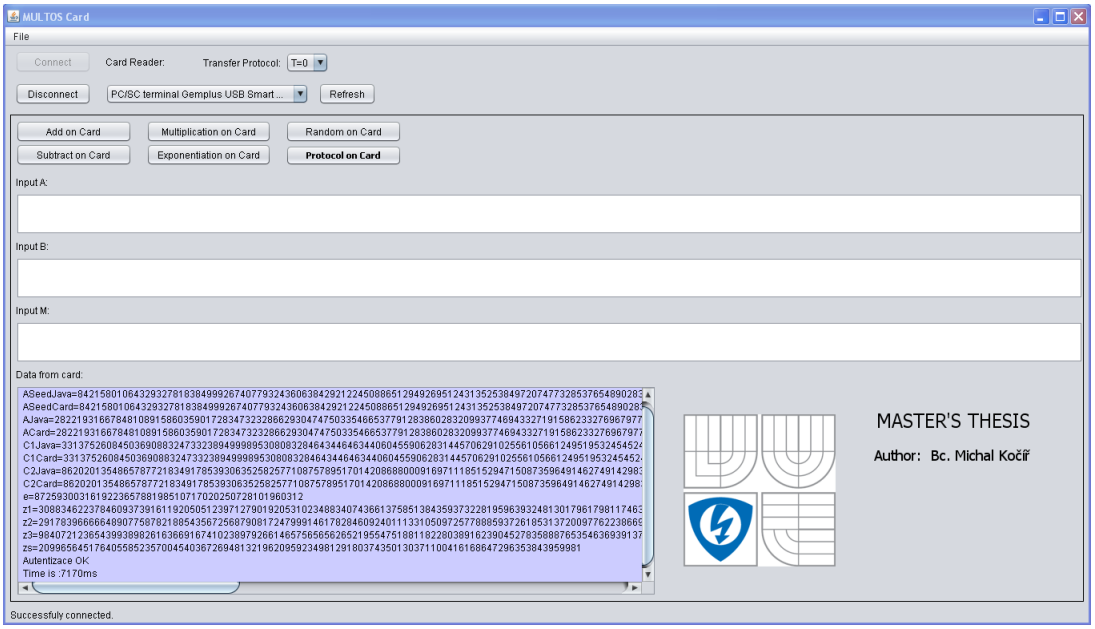
Výpočet parametru $z_1 = r_1 - eK_S w_1$:

```
ModularMultiplication(MODLENB,KA,W1,N);
ModularMultiplication(MODLENB,EA,KA,N);
SUBN(MODLENB,ZA,RA,EA);
```

Výpočet parametru $z_2 = r_2 - eK_S w_2$:

```
ModularMultiplication(MODLENJ,KB,W2,N);
ModularMultiplication(MODLENJ,EB,KB,N);
SUBN(MODLENB,ZB,RB,EB);
```

Grafické rozhraní, na kterém je znázorněn výsledek autentizačního protokolu, je zobrazeno na Obr. 5.3. Aby autentizace proběhla v pořádku, musí být výpočty jednotlivých parametrů $A_{seed}, A, C1, C2, e, z_1, z_2, z_3$ a z_s jak na kartě tak u ověřovatele stejné.



Obr. 5.3: Zobrazené grafické rozhraní.

Tab. 5.1: Doba zpracování ověření autentizace.

Parametry kryptosystému						Doba trvání autentizace mezi uživatelem a ověřovatelem pro počet opakování opn:			
Délka parametrů v bitech						opn=1	opn=3	opn=5	opn=10
$K_S[bit]$	$r_1[bit]$	$r_2[bit]$	$r_3[bit]$	$r_S[bit]$	$n[bit]$	$t_{aut}[ms]$	$t_{aut}[ms]$	$t_{aut}[ms]$	$t_{aut}[ms]$
80	480	400	640	320	1024	6950	20890	34480	68963
						6880	20670	34149	68138
						6790	20470	34099	68429
						7447	20430	34049	68057
						6546	20559	34129	68118
						6401	20460	34109	67937
						7296	20560	34129	68268
						6582	20550	34149	68027
						7177	21359	34099	68749
						6803	20828	34209	69139
Průměrná doba pro 1 atribut při opn=3; opn=5; opn=10:						6836	6894	6829	6826

V Tab. 5.1 je možné vidět, jak dlouho trvá na čipové kartě výkonnostní měření autentizačního protokolu. Uvedené časy jsou pouze orientační, protože je v nich zaznamenána i doba přenosu mezi čipovou kartou a PC.

5.5 Shrnutí

Funkční a otestovaný autentizační protokol je implementován v čipové kartě MultOS. Protokol se skládá z kryptografických operací, které jsou popsány v kapitole 4.2.

Změřená hodnota u pokročilého autentizačního protokolu se pohybuje kolem 6,8 s. Určité vylepšení času by bylo přenášet jednotlivá data mezi čipovou kartou a PC v celých blocích, tedy využívat přenosového protokolu T=1. K další nepatrnému vylepšení celkového času autentizace by mohlo dojít upravením aplikace na čipové kartě, kdy by byly využity určité výpočty ve více částech zdrojového kódu.

6 ZÁVĚR

Po nastudování funkčnosti čipových karet MultOS a teoretických částí, které jsem popsal v diplomové práci, jsem začal implementovat samotné pokročilé autentizační schéma založené na anonymní autentizaci. Toto autentizační schéma je vyvíjeno na FEKT VUT v Brně. Celé autentizační schéma jsem tedy implementoval do čipové karty MultOS a následně provedl jeho výkonnostní měření. Vzhledem k tomu jsem zároveň musel vytvořit i grafické rozhraní v programovacím jazyku Java, které je implementované v PC.

Autentizační schéma, které je vyvíjeno na FEKT VUT v Brně, umožňuje uživateli anonymní autentizaci s prokázáním osobních údajů. Je realizováno vhodnými protokoly a schématy, především se jedná o identifikaci na konceptu nulové znalosti a závazkových schématech, které jsou založeny na kryptografických primitivech.

V diplomové práci jsou tedy nejprve popsány typy autentizací, a to především autentizace předmětem - čipovou kartou a autentizace znalostí, prokázání určitého hesla popřípadě pinu. Dále následuje popis kryptografických metod. Některé jsou využity i pro implementaci autentizačního schématu, jako jsou kryptografické prvky (hašovací funkce), symetrické kryptosystémy, asymetrické kryptosystémy a kryptografické prvky v anonymní autentizaci. Další kapitola je věnována čipovým kartám se zaměřením na operační systém MultOS. V diplomové práci jsem taky zkráceně popsal analýzu platforem programovatelných čipových karet, tedy jejich srovnání jak z hlediska parametrů, tak z hlediska kryptografických operací na čipových kartách.

V praktické části diplomové práce jsem implementoval komunikaci mezi uživatelem (čipovou kartou) a ověřovatelem (vytvořená platforma na PC). Výstupem praktické části je šest samostatných aplikací umožňujících generování náhodného čísla, součet a rozdíl velkých čísel, dále pak modulární násobení, modulární umocňování a především autentizační protokol, který je vyvíjen na FEKT VUT v Brně [15].

Jednotlivě vytvořené aplikace byly otestovány a zároveň byla změřena orientační doba výpočtu jednotlivých aplikací na čipové kartě MultOS. Změřené hodnoty základních aritmetických primitiv jsou zobrazeny v Tab. 4.1. Změřená hodnota u pokročilého autentizačního protokolu se pohybuje kolem 6,8 s. Určité vylepšení času by bylo přenášet jednotlivá data mezi čipovou kartou a PC v celých blocích, tedy využívat přenosového protokolu T=1. K další nepatrnému vylepšení celkového času autentizace by mohlo dojít upravením aplikace na čipové kartě, kdy by byly využity určité výpočty ve více částech zdrojového kódu.

LITERATURA

[1] BICHSEL, P., CAMENISCH, J., GROß, T., SHOUP, V,: *Anonymous credentials on a standard java card.* In: Proceedings of the 16th ACM conference on Computer and communications security. pp. 600-610. CCS '09, ACM, New York, NY, USA (2009)

[2] BONECH, D., BOYEN, X., SHACHAM, H.: *Short group signatures.* In: Advances in Cryptology - CRYPTO'04 (2004)

[3] BURDA, K.: *Bezpečnost informačních systémů.* Online: `http://www.vutbr.cz/elearning/mod/resource/view.php?id=188856` publikováno 01. 11. 2005, Brno, [cit. 2012-10-28].

[4] CAMENISCH, J., et Al.: *Specification of the identity mixer cryptographic library,* 2010. Technical report

[5] CAMENISCH, J., STADLER, M.: *Proof systems for general statements about discrete logarithms.* Technical report, (1997)

[6] CRAMER, R.: *Modular Design of Secure, yet Practical Cryptographic Protocols.* Ph.D. thesis, University of Amsterdam (1996)

[7] CRAMER, R., DAMGARD, I., MACKANZIE, P.: *Efficient zero-knowledge proofs of knowledge without intractability assumptions.* In: Public Key Cryptography, Lecture Notes in Computer Science, vol. 1751, pp. 354-373. Springer Berlin / Heidelberg (2000)

[8] CRAMER, R., DAMGARD, I., SCHOENMAKERS, B.: *Proofs of partial knowledge and simplified design of witness hiding protocols.* In: Advances in Cryptology - CRYPTO 94, Lecture Notes in Computer Science, vol. 839, pp. 174-187. Springer Berlin / Heidelberg (1994)

[9] DAMGARD, I., FUJISAKI, E.: *A statistically-hiding integer commitment scheme based on groups with hidden order.* In: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology. pp. 125-142. ASIACRYPT '02, Springer-Verlag, London, UK(2002)

[10] DOSTÁLEK, L., VOHNOUTOVÁ, M.: *Velký průvodce infrastrukturou PKI a technologií elektronického podpisu.* [s.1.]: Computer Press, a.s., 2006. 534 s. ISBN 80-251-0828-7.

[11] HAJNÝ, J.: *Anonymous Authentication for Smartcards.* Radioenginnering. Online: http://www.radioeng.cz/fulltexts/2010/10_02_363_368.pdf 2010, -,s. 1-7. ISSN 1210-2512. [cit. 2013-04-13].

[12] HAJNÝ, J., MALINA, L.: *Accelerator Modular Arithmetic for Low-Performance Devices.* In: 34th International Conference on Telecommunications and Signal Processing. pp. 131-135. IEEE (2011)

[13] HAJNÝ, J., MALINA, L.: *Performance Evaluation of Modern Cryptographic Primitives on Current Smart-cards and Smart-phones.* Brno, 2013.

[14] HAJNÝ, J., MALINA, L.: *Practical revocable anonymous credentials.* In: Communications and Multimedia Security, Lecture Notes in Computer Science, vol. 7394, pp.211-213. Springer Berlin Heidelberg (2012)

[15] HAJNÝ, J., MALINA, L.: *Unlinkable Attribute-Based Credentials with Practical Revocation on Smart-Cards.* In Smart Card Research and Advanced Applications. Lecture Notes in Computer Science. Online: http://cardis.iaik.tugraz.at/proceedings/cardis_2012/CARDIS2012_5.pdf 2013, -,s. 62-76. ISBN: 978-3-642-37287- 2. ISSN: 0302- 9743. [cit. 2013-05-18].

[16] CHAUM, D.: *Security without identification: transaction systems to make big brother obsolete.* Commun. ACM 28, 1030-1044 (October 1985)

[17] : *Id-one cosmo v7.0.* Technical report, French Network and Information Security Agency (Agence nationale de la sécurité des systémes d'information (ANSSI)) . Online: http://www.ssi.gouv.fr/IMG/certificat/anssi-cc-cible_2009-36en.pdf 2019. [cit. 2013-04-20].

[18] JAVA CARD TECHNOLOGY. *Oracle* [online]. 2013 [cit. 2013-04-20]. Online: http://www.oracle.com/technetwork/java/javacard/downloads/index.html

[19] JURAS, S.: *Autentizace pomocí smartkaret.* Online: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=28142 publikováno 26. 05. 2010, Brno.

[20] KMENT, V.: *Hašovací funkce: Jak se odolává hackerům.* Online: http://www.lupa.cz/clanky/hasovaci-funkce-jak-se-odolava-hackerum/ publikováno 2005, [cit. 2012-11-04].

[21] MALINA, L.: *Ochrana soukromí na Internetu.* Online: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=26637 publikováno 26. 05. 2010, Brno.

[22] MAOSCO LIMITED *MULTOS Developer's Guide.* MAO-DOC-TEC-005 v1.37. 2012 Online: `http://www.multos.com/uploads/MDG.pdf` [cit. 2012-11-16].

[23] MAOSCO LIMITED *MULTOS Standard C-API.* MAO-DOC-TEC-016 v1.02. 2012 Online: `http://www.multos.com/uploads/CAPI.pdf` [cit. 2012-11-21].

[24] MULTOS CARD. *MultOS* [online]. 2013 [cit. 2013-04-20]. Online: `http://multos.com/`

[25] NAUMANN, I., HOGBEN, G.: *Enisa: Privacy features of eid cards.* Network Security Newsletter 2008, 9-13 (2008)

[26] .NET CARD. *Gemalto* [online]. 2013 [cit. 2013-04-20]. Online: `http://www.gemalto.com/products/dotnet_card/`

[27] NIST: *Gemxpresso r4 e36/e72 pk - multiapp id 36k/72k - top im gx4.* Online: `http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp771.pdf` 2009. [cit. 2013-04-20].

[28] OKAMOTO, T., UCHIYAMA, S.: *A new public-key cryptosystem as secure as factoring.* In: Advances in Cryptology - EUROCRYPT 98, Lecture Notes in Computer Science, vol. 1403, pp. 308-318. Springer Berlin / Heidelberg (1998)

[29] PINKAVA, J.: *Nové směry v kryptografii s veřejným klíčem.* Online: `http://www.lupa.cz/clanky/hasovaci-funkce-jak-se-odolava-hackerum/` publikováno 2003, [cit. 2012-05-06].

[30] RANKL, W., EFFING, W.: *Smart Card Handbook.* Munich: John Wiley & Sons, 2010. 4. ISBN 978-0-470-74367-6.

[31] SAVU, L.: *Signcryption Scheme Based on Schnorr Digital Signature.* International Journal of Peer to Peer Networks. Online: `http://airccse.org/ijp2p/papers/3112ijp2p01.pdf` 2012, -,s. 1-10. [cit. 2013-04-20].

[32] Smart Card Basis: *Smart Card Overview.* Online: `http://www.smartcardbasics.com/smart-card-overview.html` 2005. [cit. 2013-04-26]

[33] Smart Card Basis: *Types of Chip Cards.* Online: `http://www.smartcardbasics.com/smart-card-types.html` 2005. [cit. 2013-04-26]

[34] STALLINGS, W.: *Cryptography and Network Security: Principles and Practice (5th Edition).* USA: Prentice Hall, 2010. 744 s. ISBN 0136097049.

[35] – *Šifry a odesílání zpráv.* Online: http://www.palba.cz/viewtopic.php?t=5238 publikováno 22. 05. 2012, Brno, [cit. 2012-10-28].

[36] ŠÁNEK, J.: *Bezpečnost elektronických dokladů.* Online: http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=56437 publikováno 10. 06. 2012, Brno, [cit. 2012-11-29].

[37] Zive Computer: *Kryptografie,kódování dat.* Online: http://zaachi.blog.zive.cz/2008/08/kryptografie-kodovani-dat/ 2002. [cit. 2013-04-26].

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AAM Application Abstract Machine

AAS Anonymous Authentication Systems

AASR Anonymous Authentication with Spread Revelation

AES Advanced Encryption Standard

APDU Application Protocol Data Unit

API Application Programming Interface

ATR Answer To Reset

CA Certifikační Autorita

CBC Cipher-Block CHaining

CDH Computational Diffie-Hellman

CRC Cyclic Redundancy Check

CRL Certificate Revocation List

CTR Counter

DES Data Encryption Standard

DDH Decision Diffie-Hellman

DF Dedicated File

DH Diffie-Hellman

DH Discrete Logarithm

DSA Digital Signature Algorithm

DSP Digital Signal Processing

ECB Electronic Codebook

EEPROM Electrically Erasable Programmable ROM

EF Elementary File

EMV Zkratka označuje alianci Europay International, MasterCard International a Visa International spojenou s vytvořením a podporou relevantních specifikací.

f_{vz} vzorkovací kmitočet

ICC Integrated Circuit Card

IDEA International Data Encryption Algorithm

IF Integer Factorization

MAC Message Authentication Code

MD5 Message-Digest algorithm 5

MF Master File

NIPDLE Non-Interactive Proof of Discrete Logarithm Equivalency

NIZK Non-Interactive Zero-Knowledge

PK Public Key

PKI Public Key Infrastructure

RA Registrační Autorita

RAM Random Access Memory

ROM Read-Only Memory

SF Select File

SHA-1 Secure Hash Algorith

TPDU Transport Protocol Data Unit

USB Universal Serial Bus

ZKIP Zero-Knowledge Interactive Proof system

ZKPK Zero-Knowledge Proof of Knowledge

SEZNAM PŘÍLOH

A ZDROJOVÝ OBSAH PŘILOŽENÉHO DVD

A.1 Zdrojový soubor protokol.c

Konstanta utentizačního protokolu:

```
case INS_GET_AUTHENTICATE:
 if (!CheckCase(NODATAIN_DATAOUT))
  ExitSW(ERR_ISOCASE);
```

Generování náhodného čísla $K_S \in_R \{0,1\}^{80}$ o velikosti 80 bitů:

```
random(KS,MODLENK);
```

Nastavení velikosti K_S: KA(480 bitů), KB(400 bitů), KC(640 bitů), KD(320 bitů), K(1024 bitů), KE(1024 bitů)

```
memset(KA,0x00,MODLENB - MODLENK);
memcpy(KA + (MODLENB - MODLENK),KS,MODLENK);
memset(KB,0x00,MODLENJ - MODLENK);
memcpy(KB + (MODLENJ - MODLENK),KS,MODLENK);
memset(KC,0x00,MODLENA - MODLENK);
memcpy(KC + (MODLENA - MODLENK),KS,MODLENK);
memset(KD,0x00,MODLENS - MODLENK);
memcpy(KD + (MODLENS - MODLENK),KS,MODLENK);
memset(K,0x00,MODLEN - MODLENK);
memcpy(K  + (MODLEN - MODLENK),KS,MODLENK);
memset(KE,0x00, MODLEN - MODLENK);
memcpy(KE + (MODLEN - MODLENK),KS,MODLENK);
```

Výpočet: $A = A_{seed}^{K_S} \bmod n$

```
ModularExponentiation(MODLEN,MODLEN,K,N,AS,A);
```

Výpočet: $C_1 = g_3^{K_S w_{RR}} \bmod n$

```
ModularMultiplication(MODLEN,KE,W,N);
ModularExponentiation(MODLEN,MODLEN,KE,N,G3,C1);
```

Výpočet: $C_2 = g_3^{K_S} \bmod n$

```
ModularExponentiation(MODLEN,MODLEN,K,N,G3,C1);
```

Generování náhodných čísel R: RA(480 bitů), RB(400 bitů), RC(640 bitů), RS(320 bitů)

```
random(RA,MODLENB);
random(RB,MODLENJ);
random(RC,MODLENA);
random(RS,MODLENS);
```

Nastavení velikosti R: R1(1024 bitů), R2(1024 bitů), R3(1024 bitů), R4(1024 bitů)

```
memset(R1,0x00,MODLEN - MODLENB);
memcpy(R1 + (MODLEN - MODLENB),RA,MODLENB);
memset(R2,0x00,MODLEN - MODLENJ);
memcpy(R2 + (MODLEN - MODLENJ),RB,MODLENJ);
memset(R3,0x00,MODLEN - MODLENA);
memcpy(R3 + (MODLEN - MODLENA),RC,MODLENA);
memset(R4,0x00,MODLEN - MODLENS);
memcpy(R4 + (MODLEN - MODLENS),RS,MODLENS);
```

Výpočet: $\bar{A_{seed}} = g_1^{r_1} g_2^{r_2} g_3^{r_3} \bmod n$

```
ModularExponentiation(MODLEN,MODLEN,R1,N,G1,POM);
ModularExponentiation(MODLEN,MODLEN,R2,N,G2,POM1);
ModularMultiplication(MODLEN,POM,POM1,N);
ModularExponentiation(MODLEN,MODLEN,R3,N,G3,AS1);
ModularMultiplication(MODLEN,AS1,POM,N);
```

Výpočet: $\bar{A} = A_{seed}^{rS} \bmod n$

```
ModularExponentiation(MODLEN,MODLEN,R4,N,AS,A1);
```

Výpočet: $\bar{C_1} = g_3^{r_3} \bmod n$

```
ModularExponentiation(MODLEN,MODLEN,R3,N,G3,C1J);
```

Výpočet: $\bar{C_2} = g_3^{rS} \bmod n$

```
ModularExponentiation(MODLEN,MODLEN,R4,N,G3,C2J);
```

Výpočet: $e = hash\left(A, \bar{A}, A_{seed}, \bar{A_{seed}}, C_1, C_2, \bar{C_1}, \bar{C_2}\right)$

```
SHA1(MODLEN,E,A);

memcpy(POM2,A1,MODLEN);
memcpy(POM2 + MODLEN,E,MODLENE);
SHA1(MODLEN + MODLENE,E,POM2);

memcpy(POM2,AS,MODLEN);
memcpy(POM2 + MODLEN,E,MODLENE);
SHA1(MODLEN + MODLENE,E,POM2);

memcpy(POM2,AS1,MODLEN);
memcpy(POM2 + MODLEN,E,MODLENE);
SHA1(MODLEN + MODLENE,E,POM2);

memcpy(POM2,C1,MODLEN);
memcpy(POM2 + MODLEN,E,MODLENE);
SHA1(MODLEN + MODLENE,E,POM2);
```

```
memcpy(POM2, C2, MODLEN);
memcpy(POM2 + MODLEN, E, MODLENE);
SHA1(MODLEN + MODLENE, E, POM2);

memcpy(POM2, C1J, MODLEN);
memcpy(POM2 + MODLEN, E, MODLENE);
SHA1(MODLEN + MODLENE, E, POM2);

memcpy(POM2, C2J, MODLEN);
memcpy(POM2 + MODLEN, E, MODLENE);
SHA1(MODLEN + MODLENE, E, POM2);
```

Nastavení velikosti e: EA(480 bitů), EB(400 bitů), EC(640 bitů), ES(320 bitů)

```
memset(EA, 0x00, MODLENB - MODLENE);
memcpy(EA + (MODLENB - MODLENE), E, MODLENE);
memset(EB, 0x00, MODLENJ - MODLENE);
memcpy(EB + (MODLENJ - MODLENE), E, MODLENE);
memset(EC, 0x00, MODLENA - MODLENE);
memcpy(EC + (MODLENA - MODLENE), E, MODLENE);
memset(ES, 0x00, MODLENS - MODLENE);
memcpy(ES + (MODLENS - MODLENE), E, MODLENE);
```

Výpočet: $z_1 = r_1 - eK_S w_1$

```
ModularMultiplication(MODLENB, KA, W1, N);
ModularMultiplication(MODLENB, EA, KA, N);
SUBN(MODLENB, ZA, RA, EA);
```

Výpočet: $z_2 = r_2 - eK_S w_2$

```
ModularMultiplication(MODLENJ, KB, W2, N);
ModularMultiplication(MODLENJ, EB, KB, N);
SUBN(MODLENB, ZB, RB, EB);
```

Výpočet: $z_3 = r_3 - eK_S w_{RR}$

```
ModularMultiplication(MODLENA, KC, WR, N);
ModularMultiplication(MODLENA, EC, KC, N);
SUBN(MODLENA, ZC, RC, EC);
```

Výpočet: $z_S = r_S - eK_S$

```
ModularMultiplication(MODLENS, ES, KD, N);
SUBN(MODLENS, ZS, RS, ES);
```

B OBSAH PŘILOŽENÉHO DVD

Složka s programy: **xkocir02**

Složka aplikace karty:

- certifikáty
- zdrojový kód **protokol.c**
- soubor .alu, .aif, .hzx

Složka aplikace PC:

- dist - **diplomka.jar**
- diplomka.zip

Programy:

- Application Registration File Generator
- SmartDeck
- MUtil

Elektronická verze diplomové práce