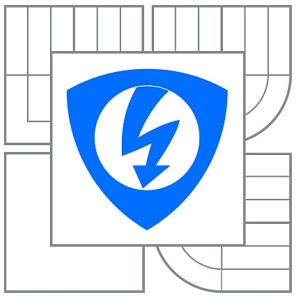
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

OPTIMALIZACE BEZPEČNOSTNÍCH ALGORITMŮ PRO SMARTKARTY

OPTIMIZATION OF SECURITY ALGORITHMS FOR SMART CARDS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

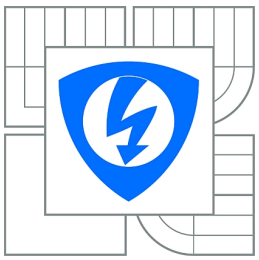
AUTOR PRÁCE — Bc. VERONIKA BARTOŇOVÁ

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN HAJNÝ

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Studentka: Bc. Veronika Bartoňová ***ID:*** 102388
Ročník: 2 ***Akademický rok:*** 2010/2011

NÁZEV TÉMATU:

Optimalizace bezpečnostních algoritmů pro smartkarty

POKYNY PRO VYPRACOVÁNÍ:

Projekt je zaměřen na seznámení se s vývojovým prostředím smartkaret (JavaCard či .NET) a na implementaci kryptografických algoritmů v tomto prostředí. Výstupem projektu je návrh komunikace mezi kartou a PC, dále pak návrh změn v kryptografických algoritmech typu prokazatelných důkazů znalosti či ověřitelných šifrování. Tyto změny by měly sloužit k budoucímu optimalizovanému nasazení algoritmů na smartkartách.

DOPORUČENÁ LITERATURA:

[1] CAMENISCH, Jan, et al Anonymous Credentials on a Standard Java Card. In . Chicago, USA : ACM, 2009. s. 600-610.

[2] MENEZES, Alfred J.; OORSCHOT, Paul C. van ; VANSTONE, Scott A. Handbook of Applied Cryptography. 2001. USA : CRC Press, 2001. 816 s. ISBN 0-8493-8523-7.

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

Vedoucí práce: Ing. Jan Hajný

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

UPOZORNĚNÍ:

ABSTRAKT

V diplomové práci se zabývám optimalizací výpočtů bezpečnostních algoritmů, které jsou dále implementovány na smartkarty. Pro návrh urychlení výpočtů vycházím ze znalosti modulární aritmetiky, která je základem pro mnohá šifrování pomocí veřejných klíčů, elektronických podpisů nebo hashovací funkce. Výsledkem projektu je návrh změn, které by měly urychlit výpočty bezpečnostních algoritmů.

KLÍČOVÁ SLOVA

Čipová karta, bezpečností algoritmy, modulární aritmetika, .NET Smart Card Framework

ABSTRACT

In this diploma thesis I dealt with proposal of optimization security algorithms calculations, that will be implemented on smartcards. This project is based on knowledge of modular arithmetic, that is used in many encryption like public keys, digital signatures and has function. The result of this project is a design of changes for accelerate the security algorithms calculations.

KEYWORDS

Smart card, security algorithms, modular arithmetic, .NET Smart Card Framework

BARTOŇOVÁ V. *Optimalizace bezpečnostních algoritmů pro smartkarty*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2011. Počet stran 59 s. Vedoucí diplomové práce byl Ing. Jan Hajný.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Optimalizace bezpečnostních algoritmů pro smartkarty“ jsem vypracovala 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 autorka uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

Ráda bych poděkovala Ing. Janu Hajnému za jeho rady a odborný dohled při zpracování diplomové práce.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

V diplomové práci se zabývám optimalizací bezpečnostních algoritmů smartkaret. Problematika smartkaret není zase tak velkou neznámou. Tento trend již našel využití v různých sférách a odvětvích, nejčastěji je možné se s ní setkat v podobě jízdenek v městské hromadné dopravě, zdravotních karet či karet pojištění ve zdravotnictví, plateb debetními kreditními kartami ve finančním sektoru nebo kontroly přístupu do chráněných prostor.

V dnešní době se považuje za primární ochránit svá data. Nejenom zabezpečit jejich úschovu, ale i manipulaci s nimi. Díky stále novým technologiím a rostoucí rychlosti Internetu je prakticky nutné vhodně zvolit takové zabezpečovací techniky aby špatná manipulace s citlivými daty nezpůsobila uživateli velké finanční ztráty. Stejnou zvyšující se rychlostí Internetu se zvyšují i nároky, které jsou kladeny na bezpečností algoritmy. Už nestačí chránit data samotná, ale principy jejich ochrany musí být co nejrychlejší a nejefektivnější. S tímto faktem souvisí i technická vybavení čipových karet. Bezpečnostní algoritmy, které čipové karty podporují jsou závislé na technickém vybavení karty - na velikostí paměti nebo procesoru. Aby nebylo nutné vyrábět stále lepší a lepší mikroprocesory, pozornost se obrátila na výpočty bezpečnostních algoritmů a jejich možné úpravy.

V rámci diplomové práce vycházím ze znalosti modulární aritmetiky, která je základem pro mnohá šifrování pomocí veřejných klíčů, elektronických podpisů nebo hashovací funkce. Mou snahou je navrhnout změny, které by měly urychlit výpočty bezpečnostních algoritmů na čipových kartách.

2 PROBLEMATIKA BEZPEČNOSTNÍCH ALGORITMŮ

Ochrana citlivých dat nebo systémů je již dlouhou dobu aktuálním tématem. Existuje nepřeberné množství bezpečnostních algoritmů, které pracují na tom či onom principu. Jedno mají, ale všechna společné, pracují na principu šifrování zprávy (textu, bitové posloupnosti).

Šifrování obecně vychází z předpokladu transformace otevřené zprávy na zprávu zašifrovanou, kde je nutné definovat šifrovací a dešifrovací klíče K_e, K_d. To zda šifrovací klíč bude stejný jako dešifrovací, záleží již na použité šifře. V systémech s tajným klíčem je šifrovací klíč stejný jako dešifrovací ($K_d = K_e$), nebo je z šifrovacího snadno odvoditelný. ($K_d = f(K_e)$). U systému s veřejným klíčem platí, že oba klíče jsou voleny tak, aby nebylo možné ze znalosti jednoho klíče odvodit druhý. Šifrování lze formulovat rovnici, kde Z představuje množinu zpráv, která má být zašifrována a C značí zašifrovaný text (kryptogram): [16]

$$C = E(Z, K_e)$$

Obrácené zobrazení (dešifrování) lze definovat rovnici, kde D označuje dešifrovanou zprávu tj. zprávu ve tvaru před samotným šifrováním:

$$Z = D(C, K_d)$$

Šifrovací klíče jsou sestavovány pomocí kryptografických výpočtů, které zvyšují bezpečnost zprávy. Díky těmto výpočtům je možné dosáhnout utajení obsahu při přenosu, integritu dat a autenticitu zprávy. Kryptografické výpočty lze rozdělit do několika skupin

1. Algoritmy symetrické kryptografie.
2. Algoritmy asymetrické kryptografie.
3. Autentizační kódy zpráv.
4. Generování náhodných a pseudonáhodných čísel.
5. Hashovací funkce.

Většina zmíněných kryptografických skupin nebo systémů čerpá z poznatků matematické algebry, převážně z aritmetiky kódu zbytkových tříd.

2.1 Aritmetika kódu zbytkových tříd

Mnohá šifrování pomocí veřejných klíčů, elektronických podpisů nebo hashovací funkce, jak již bylo zmíněno, vyžadují uplatnění aritmetiky kódu zbytkových tříd. Právě tato aritmetika umožňuje provádění výpočtů s konečnou přesností pro zobrazení nekonečných množin reálných čísel. Je nutné si uvědomit, že celkovou aproximaci reálných čísel na počítači reprezentuje konečná množina $F \subseteq R$. Tato množina bývá označována jako soustava čísel s pohyblivou řádovou čárkou nebo čísla reprezentovatelná na počítači. Množina F má následující vlastnosti [17]:

- F je konečná podmnožina reálných čísel Q.
- F je obyčejně symetrická vzhledem k počátku, kde jsou udávány dvě počítačové reprezentace nuly.
- Elementy množiny F nejsou rozloženy rovnoměrně podél reálné osy, interval mezi dvěma sousedními čísly reprezentovatelnými počítačem je velmi malý, v blízkosti nuly a s rostoucí vzdáleností od ní se zvětšuje.
- Všechna známá racionální čísla jsou z množiny F vyloučena. Např. 1/10, 1/3, 5/6 atd. Vzhledem k binární reprezentaci čísel jsou prvky množiny F jen racionální čísla typu p/q, kde q je mocnina dvou.

2.1.1 Jedno-modulová aritmetika kódu zbytkových tříd

Většina číslicových počítačů vykonává jen některé aritmetické operace přesně, jedná se především o ty, kde jsou operandy celá čísla. S ohledem na tento fakt je nutné používat celočíselné aritmetiky a výsledek poté redukovat modulem m. Samotná redukce modulem je známa pod názvem jedno-modulová aritmetika kódu zbytkových tříd (single-modulus residue arithmetic), celé číslo m (m>1) modulem aritmetického systému a symbolem Z množinou celých čísel.

Na rozdíl od klasických celočíselných operací se v modulární aritmetice provádí celočíselné dělení modulem n a výsledkem je zbytek po tomto dělení. Díky tomu, že množina Zn je konečná, jsou běžné operace jednodušší, rychlejší a potřebují konstantní množství paměti. Což umožňuje modulární aritmetiku prakticky využít nejenom ve výpočetní technice (aritmetika s velkými celými čísly, pseudonáhodná čísla), ale i pro přenos zprávy (ochrana zpráv proti chybám, komprese). Hodnota modulu je nejčastěji definována jako kladné celé číslo, které nemusí být prvočíslo. Například šifrování RSA, Rabin nebo ElGamal vyžadují efektivní metody pro provádění násobení a umocňování v Zn. [12]

Modulární aritmetika využívá kongruence a dělitelnosti. Právě tyto vlastnosti modulu n umožňují počítat pouze se zbytky po dělení tímto modulem a výsledek

poté zobecnit na všechna čísla. Množinu všech celých čísel, která jsou kongruentní s nějakým n modulo lze také označit jako třídu kongruence[1] a značit ji $\bar{a}$.

Výpočty modulární aritmetiky

Sčítání a násobení

Některé operace modulární aritmetiky jsou si velmi podobné, například sčítání a násobení. Oba dva výpočty se chovají stejně jako při sčítání a násobení v klasické aritmetice celých čísel s tím rozdílem, že výsledek je vždy redukovaný podle modula N.

1. Sčítání - $[a]modN + [b]modN = [a + b]modN$
2. Násobení - $[a]modN \cdot [b]modN = [a \cdot b]modN$

Sčítání spadá do skupiny uzavřených, komutativních a asociativních operací s jednotkovým prvkem nula. V rámci této skupiny je možné ke každému prvku a z množiny celých čísel najít inverzní prvek $b = N - a$. Násobení naopak náleží do skupiny uzavřených, komutativních, asociativních operací s jednotkovým prvkem jedna. V této skupině je počet inverzních prvků pro operaci násobení v množině celých čísel rovný funkci $\varphi(n)$ (Eulerově funkci). [12]

Umocnění

Modulární umocnění je považováno za časově náročnější aritmickou operaci a také velmi důležitou pro asymetrické kryptografické systémy. Pro výpočet mocniny lze vycházet z Eurelovy funkce $\varphi(n)$ a malé Fermatovy věty, ze které vyplývá, že $g^{\varphi(n)} \equiv 1 mod\ N$ pokud existuji nesoudělitelná taková g a n. Výpočet modulárního umocnění je poté definován takto: $c = a^e \cdot \mathrm{mod} N$. Existuje několik variant jak přistupovat k výpočtu umocnění [12]:

- využití binárního umocňování ve směru z prava/doleva nebo zleva/do prava - pro aplikaci tohoto způsobu je nutné vycházet z posloupnosti součtů[2], která je dána binárním zápisem exponentu e. Pro provedení tohoto typu stačí malá paměť na ukládání mezivýsledků. Směr zda bude umocňování z prava čí zleva určuje přístup k exponentu.

[1] Kongruence označuje ekvivalenci na algebře, kterou lze kombinovat se všemi operacemi v algebře.

[2] Posloupnost součtů je definována jako nalezení nejkratší posloupnosti součtů pro číslo e, takových aby každé z čísel bylo součtem některých dvouch předcházejících.

- využití základního k-násobné umocňování - pro využití tohoto způsobu výpočtu je nutné rozdělit exponent do bloků o velikosti k bitů a poté se vypočítají mocniny g^i pro index i od 0 do $2^k - 1$. Tohoto způsobu výpočtu se využívá v případech, kdy je nutné snížit počet provedených násobení, protože zpracovává víc než 1 bit v exponentu v jednom opakování.

- využití umocňování pomocí posuvného okna - tento způsob výpočtů využívá k-násobné umocňování, ale pro proměnnou délku okna. Exponent je rozdělen do bloků, které jsou od sebe navzájem odděleny nulou.

- využití Montgomeryho umocnění - tento výpočet využívá pro výpočet $x^e mod N$ kombinaci Montgomeryho násobení a binárního umocňování provedené zleva do prava. Montgomeryho umocňování patří mezi nejpoužívanější algoritmus pro implementaci modulárního umocňování.

V tabulce 2.2 je uveden čas pro vykonání jednotlivých operací, testy probíhaly na platformě Javacard. [12]

2.1.2 Největší společný dělitel

Dělitelnost celých čísel vychází z faktu,že celé číslo a je dělitelné celým číslo b ($c = a/b$). Pokud nastane případ, že hodnota c nebude celé číslo je nutné použít Větu dělení se zbytkem, která zní: Pokud čísla $a > 0$ a $b > 0$ poté existuje dvojice celých čísel q, r kde $q \geq 0$, $0 \leq r < a$, $b = a \cdot q + r$. V této větě představuje q podíl a r zbytek po dělení čísel a a b. Největší společný dělitel pro čísla a a b je poté celé číslo c, které dělí a a b.

Dále lze uvažovat, že největší společní dělitel celých nenulových čísel a a b je možné vyjádřit i jako součet $ma + nb$, kde m a n představují celá čísla.

2.1.3 Euklidův algoritmus

Pro výpočet největšího společného dělitele(gdc) dvou kladných celých čísel se používá také Euklidův algoritmus. Na základě tohoto algoritmu jde také vypočítat inverzní prvek nebo vyjádřit gdc dvou čísel jako součet jejich násobků.

Princip Euklidova algoritmu spočívá v nalezení takového přirozeného čísla, které dokáže čísla a a b dělit beze zbytku. Celý algoritmus funguje následovně: na vstupu se příjmou čísla a a b, v každém dalším kroku se vydělí se zbytkem číslo a číslem b. Pokud zbytek není nula, do a bude přiřazeno b a zbytek po dělení se vloží do uvolněné proměnné b a celý postup se opakuje. V daný moment, kdy zbytek po dělení je nulový, musí být v proměnné b uložen největší společný dělitel čísel a a b. [13]. Pro ilustraci výpočtu gdc je uveden příklad v tabulce 2.1.

a	b	a = b · q + zbytek
12345	1350	12345 = 9 · 1350 + 195
1350	195	1350 = 6 · 195 + 180
195	180	195 = 1 · 180 + 15
180	15	180 = 12 · 15 + 0
15		

Tabulka 2.1: Výpočet gdc (12345,1350).

Na základě rozšířené verze Euklidova algoritmu lze také spočítat i multiplikativní inverzi[3], kde největší společný dělitel dvou čísel je vyjádřen pomocí Bézoutovou rovnosti[4], tedy jako lineární kombinaci daných čísel a jejich koeficientů z řad celých čísel.

Výpočet inverze

V situaci, kdy největší společný dělitel má hodnotu 1 (gdc = 1) je možné provést výpočet jeho inverzního prvku x modulo N. Celý postup vychází z vyjádření Bézoutovy rovnice, která po doplnění hodnot vypadá takto: $ax + bN = 1$. Poté matematickou úpravou lze získat rovnici $ax = 1 \bmod N$, kde nalezené a je hledaný inverzní prvek. Tento výpočet inverzního prvku se často objevuje v aplikacích teorie čísel.

2.1.4 Vlastnosti prvočísel

Označení prvočíslo definuje takové přirozené číslo, které je beze zbytku dělitelné dvěma různými přirozenými čísly, z toho jedno z nich musí být číslo 1 a druhé původní zadané číslo. Mezi nejznámější prvočísla patří především 3,5,7,11 aj. Zkoumáním vlastností se zabývá teorie čísel. [14]

Vlastnosti prvočísel

1. Každé složené číslo[5] je možné vyjádřit jako součin prvočísel - faktorizace (například $28 = 2^2 \cdot 7$).

2. P bude klasifikováno jako prvočíslo právě když bude platit Wilsonova věta $(p-1)! \equiv -1(\bmod p)$.

3. Suma převrácených hodnot prvočísla diverguje.

[3]Multiplikativní inverze čísla $x \in Z$ je takové číslo $x - 1$, pro které platí, že $x * x - 1 \equiv 1$.
[4]Bézoutova rovnice je dána vztahem $gdc(a, b) = \alpha a + \beta b$ kde $a, b \in N$ a $\alpha\beta \in Z$.
[5]Složené číslo označuje takové číslo, které není provočíslo

4. Jestliže platí, že G je konečná grupa s p^n prvky, tak také obsahuje G prvek řádu p.

5. Za podmínek, že $n > 1$ je kladné celé číslo, poté existuje p tak, že $n < p < 2n$ (podmínka vychází z Bertrandova postulátu).

6. Pokud p je prvočíslo a platí, že a $0 < a < p$ je celé číslo poté $a^p - a$ je dělitelné p.

7. Pokud je p prvočíslo a p dělí součin čísel $(a \cdot b)$, poté p dělí a nebo p dělí b.

Testy prvočíselnosti

Testy prvočíselnosti se provádí pro ověření zda zadané číslo může být prvočíslem.

1. Elementární test prvočíselnosti - oblíbený velmi jednoduchý algoritmus, zkouší u testované čísla všechny jeho možné dělitele, neefektivní pro práci s velkými čísly.

2. Rabin-Millerův test - tento test je založen na myšlence rovností, které jsou splněny pro prvočísla, ale obecně neplatí, platnost je vždy zkoušena pouze na testovaném čísle. Rabin-Millerův test stanovuje zda číslo s pravděpodobnosti $1 - 1/4^k$ je prvočíslo, když k je počet iterací testu.

3. Lehmannův test.

4. Fermatův test - vychází z malé Fermantovy věty[6], která říká, že pro každé prvočíslo p nesoudělné s číslem $a \in Z_p$ platí $a^{p-1} \equiv 1$. Díky ní je možné vypočítat nejenom nejmenší kladný zbytek ale i multiplikativní inverzi.

Výpočet nejmenšího kladného zbytku

Při hledání nejmenšího kladného zbytku například pro 5^{203} modulo 101 je nutné vycházet z malé Fermantovy věty, což znamená, že platí $5^{100} \equiv 1($ modulo 101). Poté výpočet kladného zbytku bude vypadat následovně:

$5^{203} = (5^{100})^2 \cdot 5^3 \equiv 125 \equiv 24($ modulo 101).

[6]Malá Fermatova věta je základním stavebním kamenem algoritmu generovaní šifrovacího klíče asymetrické šifry RSA. Je také nutnou podmínkou pro prvočísla. [13]

	256 bitů	512 bitů	1024 bitů	2048 bitů
Sčítání	0,257 μ s	0,429 μ s	1,101 μ s	2,330 μ s
Násobení	9,329 μ s	49,749 μ s	153,714 μ s	640,025 μ s
Dělení	16,018 μ s	77,576 μ s	253,098 μ s	1009,711 μ s
Umocnění	9,428 ms	94,275 ms	707,205 ms	5224,972 ms

Tabulka 2.2: Čas potřebný pro výpočet aritmetických operaci

2.1.5 Zápis a reprezentace velkých čísel

Sada registrů v dnešních procesorech má velikost 32 nebo 64 bitů, což znamená že největší binární číslo se kterým může počítač bez větších problému pracovat je $2^{32}(2^{64})$. Pro čísla, která jsou větší je nutné použít vhodnou reprezentaci. Pro zápis velkých čísel se obyčejně používají poziční číselné soustavy[7]. V každé poziční soustavě je možné vyjádřit číslo s pomocí rozvoje následovně:

- $s = a_n b^n + a_{n-1} b^{n-1} + ... + a_1 b^1 + a_0 b^0$ nebo zkráceně $s = (a_n a_{n-1}..a_1 a_0)_b$
  kde $b \in N/1; a_i \in 0..b-1$

Označení jednotlivých použitých prvků v rovnicích: a představuje číslice (cifru), index i řád číslice, b bázi (základ) číselné soustavy, a_n číslice nejvyššího řádu naopak a_0 určuje číslici nejnižšího řádu. Mezi nejpoužívanější číselné soustavy patří ty se základem 10, 6 a 2. Šifry jako RSA, Rabin nebo ElGamal využívají při svých výpočtech velké celé číslo, v řádu několika stovek bitů. Takto velká čísla není možná uložit do datových typů dostupných na současných počítačích. Proto se přistupuje k reprezentaci v již zmíněné poziční soustavě a jejímu zápisu. Ve většině případu bází číselné soustavy je jednoduchý datový typ ve většině případu o velikosti 1 byte. Samotné hodnota velkého čísla je uchována v datové struktuře typu jednoduchého datového pole. Velikost čísla uložené tímto způsobem může být omezena množstvím dostupné paměti. [12]

2.2 Šifry založené na modulární aritmetice

Výše uvedené matematické operace se používají v rámci šifrování. V následujícím odstavci jsou uvedeny příklady šifer, které jsou založeny na modulární aritmetice.

[7] Jedná se o druh číselné soustavy, kde je hodnotě každé číslice dána její pozice v sekvenci symbolů.

2.2.1 Znakové šifry

Za každou složitější šifrou stojí šifry jednoduché, založené na substituci, transpozici či heslu (aditivní šifry). Klíčem k rozluštění substitučního systému je substituční tabulka se znaky otevřeného textu, kterým přiřazuje znaky šifrovaného textu. Typickým příkladem substituční šifry může být Caesarova šifra, kde je každému písmenu přiřazeno písmeno, které je o tři místa posunuté doprava. Tedy písmeno A je převedeno na D, B na E apod.

Caesarova šifra může být chápana i jako aditivní systém, protože při posunutí dochází k přičítání posloupnosti C (C je v abecedě ztotožněno hodnotou 3). U složitějších aditivních - heslových systémů jako je například Vigenérova šifra je heslo tvořené periodicky se opakujícím klíčovým slovem. Transpoziční systémy jsou na rozdíl od substitučních či aditivních znaky otevřeného textu nemění, jen je přeskupuje. Mění jejich pořadí v rámci slov či vět. Uvedené šifry patří do skupiny tzv. symetrických šifer, kde pro šifrování a dešifrování je použit stejný klíč.

2.2.2 Blokové šifry

Pro nesnadné rozšifrování textu na základě zjištění četnosti výskytu písmen byl vyvinut systém blokových šifer, který nahrazuje bloky určité délky otevřeného textu bloky stejné délky textu šifrovaného. Každý blok otevřeného textu se převádí jako nejmenší nezáporný zbytek modula N lineární kombinací prvků v daném bloku. Díky faktu, že blokové šifry pracuji s bloky písmen nikoliv s jednotlivými písmeny, nejsou zranitelné kryptoanalýzou založenou na frekvenční metodě.

Za to kryptoanalýza blokových šifer s použitím výskytu četnosti jednotlivých kombinací písmen v bloku o velikosti n má smysl pouze bude-li n malé číslo. Například, v případě kdy $n = 10$, existuje $26^{10} = 1,4\text{x}10^{14}$ kombinací písmen s touto délkou. Pro takto velký počet je velmi obtížné použít analýzu založenou na srovnání relativních četností daných kombinací v šifrovaném textu s výsledky relativní četnosti získanými z textu obyčejného. [17]

2.2.3 Exponenciální šifry

Exponenciální šifry využívají modulární umocňování, k zašifrování zprávy je nutné převést text zprávy do dvojciferného číselného tvaru a tato čísla poté seskupit do $2s$ bloků dekadických číslic, kde s je největší kladné sudé celé číslo. Poté každý blok otevřeného textu se převede na blok šifrovaného textu dle vzorce $c = |p^e|_m$, kde p udává blok otevřeného textu a e představuje šifrovací klíč. Pro dešifrování bloku c je nutné znát dešifrovací klíč tj. takové celé číslo d - multiplikativní inverzi

$e\ mod(m-1)$. Exponenciální šifra se především používá pro vytvoření společného klíče pro dva nebo více subjektů. [17]

2.2.4 Kryptografie eliptických křivek

Kryptografie eliptických křivek (ECC) představuje další alternativu pro šifrování i pro elektronický podpis. ECC má výhodu v rychlosti a menší náročnosti na zvolený hardware. Eliptické křivky si lze představit jako množiny bodů v rovině, pro kterou platí rovnice $y^2 = x^3 + ax + b$. Princip šifrování a a podepisování v ECC je založen na problému tzv. diskrétního algoritmu, kde je nutné najít z dvou bodů P a Q, které jsou součástí veřejného klíče, takové číslo, aby platila rovnice $Q = kP$. [17]

3 PROBLEMATIKA SMARTKARET

3.1 Základní definice

Smartkarta neboli čipová karta, je speciální typ plastové karty, která disponuje integrovaným obvodem/čipem, díky němuž dokáže přijmout a zpracovat data a poté vrátit požadované informace. Mezi její další přednosti patří spolehlivý implementovaný systém zabezpečení, který obsahuje nejenom ochranu proti neoprávněné manipulaci s daty, či jejich falšování, ale i důkladné zabezpečení čipu proti cizímu přístupu.

Smart karty lze obecně rozdělit podle typu vestavěného čipu a metody komunikace se čtecím zařízením.

1. Typy vestavěných čipů

   - Paměťové - jsou přizpůsobeny k ochraně informací. Paměťový prostor na těchto kartách může být zabezpečený pouze pro čtení a zápis a nebo nezabezpečený - volně přístupný.
   - Mikroprocesorové - jsou uzpůsobeny pro umístění zabezpečení proti neautorizovaném převodu, psaní a čtení. Mikroprocesorové čipové karty obsahují programy a OS, které se využívají pro převod dat pomocí algoritmů pro funkčnost těchto algoritmů je umístěn na čipu i patřičný hardware: mikrořadič, centrální procesor, koprocesor, operační paměťové zařízení a stálý paměťový prostor. Fyzické konstrukce je nakreslena na obrázku 3.1.

2. Metody komunikace s čtecím zařízením

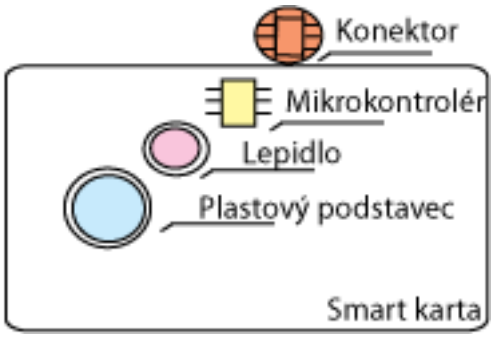
   - Kontaktní - pro komunikaci je nezbytné čipovou kartu vložit do čtecího zařízení, aby došlo k propojení vodivých kontaktů. Čipová karta obsahuje kontaktní plochu s osmi kontakty. Každý z kontaktů má různou funkci (viz obrázek 1). V dnešní době se vyrábí karty integrující USB rozhraní přímo na čipu (označované USB-ICC), což umožňuje nahradit čtečky čipových karet USB rozhraním počítače. K počítačovému rozhraní je poté ještě připojen kontaktní adaptér s čipovou kartou v formátu SIM[1]. [5]
   - Bezkontaktní - tzv. bezdotykový přístup k informacím. Pro komunikaci je nutné přiložit kartu ke čtecímu zařízení. Komunikace s čipovou kartou probíhá pomocí rádiového rozhraní na krátkou vzdálenost (od 10 cm)

[1]SIM - normovaná velikost dle ISO/IEC 7810 je 25x15mm.

využitím dané frekvence[2]. V čipové kartě je zabudováno bezkontaktní rádiové rozhraní fungující na principu elektromagnetické indukce a přenosu informací elektromagnetickým polem.

- S dvojím rozhraním - karta je vybavena dvěma čipy, kontaktním i nekontaktním.

Čipové karty jsou normovány mezinárodními standardy. Norma ISO/IEC 7816 obsahuje standardy pro kontaktní čipové karty v šestnácti částech. Za zmínku stojí norma ISO/ICE 7816-1, ve které jsou definovány fyzické rozměry čipových karet, jejich odolnost při statické elektřině a fyzická odolnost při ohýbání karty [1]. Další norma ISO/IEC 7816-12 výrazně rozšířila možnosti komunikace kontaktních čipových karet pomocí USB rozhraní a ISO/IEC 7816-15 definovala kryptografické informace a mechanismy. Norma ISO/IEC 1443 definuje bezkontaktní čipové karty ve čtyřech částech. Jednotlivé části slouží pro definici základních fyzikálních charakteristik, radiofrekvenčních výkonových a signálových rozhraní, inicializací spolu s anti-kolizí a přenosového protokolu. [5]



Obrázek 3.1: Fyzická konstrukce mikroprocesorové čipové karty

3.2 Technické parametry smartkaret

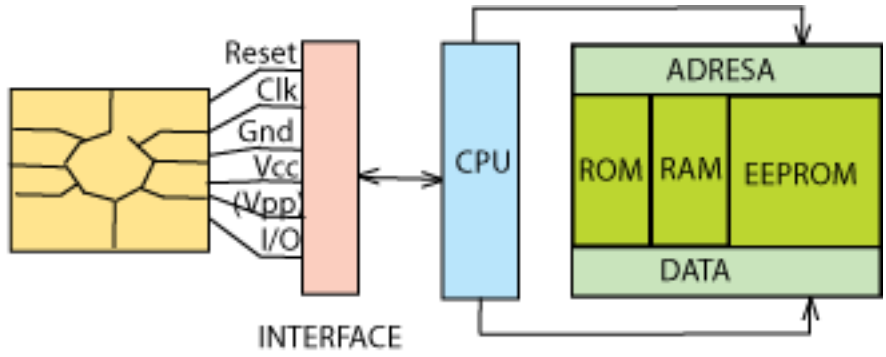
Jak již bylo uvedeno, kontaktní čipová karta je spojena s vnějším světem plochou o osmi kontaktech. Toto schéma je možné vidět na obrázku 3.2. Mimo kontaktní plochy obsahuje karta ještě mikroprocesor a paměti. Na kartě je celá paměť situována do několika oblastí:

- Oblast typu RAM - tato paměť při manipulaci s kartou ztrácí svůj obsah (vyjmutí ze čtecího zařízení). Využívá se pouze pro uložení dočasných výsledků při výpočtech.

[2]Dle ISO/EIC1443 se používá frekvence 13,56 MHz.

- Oblast typu ROM - tato paměť obsahuje operační systém karty. Kapacita této paměti bývá nejčastěji několik desítek kilobajtů. Paměť tohoto typu nelze přepsat.

- Oblast typu EEPROM - tato paměť si oproti RAM paměti pamatuje informace i po vyjmutí karty ze čtecího zařízení a obsahuje důležitá data pro provoz aplikace. Dále je možné tuto paměť naprogramovat nebo smazat. Správné fungování paměti typu EEPROM je závislé na její kapacitě, řádově se využívá kapacita nepřevyšující desítky KB.



Reset - kontakt pro reset mikroprocesoru
Clk - kontakt pro hodinový signál
Gnd - kontakt pro připojení referenčního napětí
Vcc - kontakt pro napájení mikroprocesoru
Vpp - programovací napěťový vstup
I/O - vstup a výstup pro sériový tok dat

Obrázek 3.2: Struktura kontaktní procesorové čipové karty

Operační systém čipové karty implementuje příkazy komunikačního rozhraní APDU[3] a plní funkci i jako management dat na kartě. Podle OS implementovaného výrobcem lze přesně určit, k čemu je možné čipovou kartu použít. S ohledem na vlastnosti OS lze čipové karty rozdělit na statické nebo dynamické čipové karty.

- Statické - karty (označované jako statické) umožňují čtení dat, jejich zápis a práci s nimi. Do této skupiny lze zařadit SIM karty[4], Cryptoflex, StarCos, Gemplus).

[3]APDU - Aplication Protocol Data Unit - jedná se o jednotný formát zpráv definovaný nad protokoly T=0 a T=1. [4]

[4]SIM - Subscriber Identity Module - identifikanční číslo účastníka v mobilní síti.

- Dynamické - na karty tohoto označení lze nahrát programový kód a jeho prostřednictvím něho měnit chování karty. V současné době mezi nejrozšířenější patří JavaCard nebo Gemalto .NET Card.

Organizace dat na kartě probíhá pomocí souborové systému. Jedná se převážně o uložení citlivých dat a případnou práci s nimi. Souborový systém je obdobou struktury dat na disku PC - data se ukládají do adresářů podle důležitosti [4].

- MF - Master File = kořenový adresář / disk.
- DF - Dedicated File = adresář aplikace.
- EF - Elementary File = datový soubor.
- Speciální soubory - jedná se o EF obsahující data, která jsou zprostředkována OS karty (soukromé klíče apod.).

Data se na čipové kartě ukládají jako objekty s atributy. Kupříkladu soukromý klíč není uložen pouze v jednom datovém souboru, ale i v několika dalších, kde jsou uloženy informace o vlastních datech, příznaku soukromého klíče, nebo o použitém algoritmu soukromého klíče. Aplikace přistupující k datům se v jednotlivých APDU příkazech odkazuje na dané atributy nikoliv na celá data. Atributem objektu se může stát i příznak v případě, že se jedná o soukromý či veřejný objekt čipové karty.

Komunikace s čipovou kartou probíhá přes komunikační rozhraní. Pro toto rozhraní jsou navrženy dva přenosové protokoly T=0 a T=1[5]. T=0 patří do skupiny znakově orientovaných protokolů, dále odděluje také požadavky a odpovědi na ně. T=1 protokol je na rozdíl od T=0 orientovaný blokově a požadavky ani odpovědi na ně neodděluje. Komunikace s nekontaktní čipovou kartou probíhá pomocí protokolu T=CL. Struktura APDU je k dispozici na obrázku 3.3. Samotný přenos informací může poté probíhá na dvou rozhraních na APDU a na API[6].

1. APDU rozhraní - jedná se o společné rozhraní pro všechny typy čipových karet, které tímto způsobem vylučují nekompatibilitu mezi nimi nebo mezi různými OS.
2. API rozhraní - toto rozhraní komunikuje s kartou detailně určeným rozhraním za použití ovladače (PKCS#11[7], CSP[8] a jiné.) pro daný typ karty. Ovladače neboli knihovny pro komunikaci s kartou poskytuje jejich výrobce. [4]

[5]T=0 a T=1 jsou standardizované přenosové protokoly dle normy ISO/IEC 7816-3.

[6]API - Application Interface - jedná se programovatelné rozhraní aplikace.

[7]PKCS#11- jedná se standart použitelné pro rozhraní s kryptografickými tokeny.

[8]CPS - Cryptographic Service Provider - knihovna v Microsoft Windows, která umožňuje kódování a dekódování funkcí.

CLA = třída instrukce
INS = vlastní instrukce
P1, P2 = parametry instrukce
DATA = v obou případech transportovaná data

Lc = velikost dat předávaných do karty v rámci příkazu v bytech
Le = velikost dat očekávaných v odpovědi v bytech
SW1, SW2 = návratové kódy příkazu, SW2 definováno v závislosti na SW1. Pokud se použije komunikační protokol T=0 jsou posílaný pouze SW1,2 data jsou posílána samostatně.

Obrázek 3.3: Struktura APDU

3.3 Zajištění bezpečnosti dat

Smartkarty obecně umožňují dvou-faktorovou bezpečnost, tzn. bez znalosti přístupového hesla (PIN) a čipové karty není možné získat přístup k důvěryhodným informacím. PIN má vždy čtyřmístný tvar a je složen z číslic tj. 10 000 možných kombinací. S ohledem na tento počet umožňují čipové karty také omezení počtu pokusů o vložení PIN. Po jejich vyčerpání se karta ve většině případů zablokuje nebo zničí. [2]

Díky tomu, že majitel čipové karty má svou kartu vždy u sebe, zabraňuje nebezpečným útokům na ni. Útoky na čipové karty jsou znesnadněny i strukturou karty(obsahují jediný čip), kdy nelze sledovat komunikaci mezi jednotlivými prvky. Navíc současné čipové karty jsou odolné i vůči sofistikovaným útokům (jako jsou slovníkové útoky) a v krajním případě dokáží zničit uložená data. [6]. Mimo PIN lze zabezpečit čipové karty i o další bezpečnostní prvky například o magnetické proužky nebo natištěné informace (identifikační číslo, čárový kód).

V dnešní době patří kontaktní čipové karty mezi nejuniverzálnější a v bezpečnosti k nejpoužívanějším. Tyto karty mohou podporovat i elektronickou autorizaci dat (elektronický podpis), PKI[9] a šifrování (asymetrické nebo symetrické). Díky

[9]PKI - Public Key Infrastructure, jedná se o označení správy a distribuce veřejných klíčů v asymetrické kryptografii.

mikroprocesoru lze provádět složité výpočty, mezi které často patří zašifrování a dešifrování dat pomocí klíče uloženého na kartě, což rozšiřuje možnosti jednoznačného určení uživatele přistupujícího k systému. [4]

3.3.1 Elektronická autorizace dat

Elektronickým podpisem se rozumí digitální podpis ověřitelný s použitím veřejného klíče, jenž je součástí certifikátu[10]. Vznikl za účelem přenesení rukou psaného podpisu do světa moderních technologií. Nejčastěji je možné se s tímto prvkem setkat při nalogovaní do zabezpečeních aplikací využívající protokol https (internetové bankovnictví, poštovní klienti aj.). Princip digitálního podpisu je založen na jedinečném páru veřejného a soukromého klíče. Tento princip vychází z asymetrického šifrování, které se vyznačuje tím, že pro zašifrování dat je nutné použít jiný klíč než pro šifrování, což je případ digitálního podpisu. Základem digitálního podpisu je jedinečný pár klíčů, u kterých je předem dané zda se jedná o klíč veřejný nebo soukromý. Samotný digitální podpis poté vzniká tak, že data určená k podepsání jsou převedena pomocí hash funkce na zkrácený řetězec znaků a digitální podpis je připojen k původním nešifrovaným datům. Veřejný klíč poté slouží k ověření, zda daný digitální podpis vznikl použitím párového soukromého kílče použitého na daný text. [8]

Pro digitální podpis se využívají asymetrické algoritmy RSA, DSA. U certifikační autority používá privátní klíč délku 1024 bitů. Délky RSA klíčů 2048 bitů a 4096 bitů se v tuhle chvíli používají pro digitální podpis serverů a certifikačních autorit samotných. [3]

V souvislosti s čipovou kartou elektronický podpis probíhá následně. Potřebné klíče se vygenerují přímo na čipu karty za použití metod pro generování náhodných čísel, kde také zůstávají. Čipovou kartu není možné zcela zkopírovat a v tuto chvíli neexistuje funkce, která by umožnila získání tajného klíče. Při použití karty data určená pro podpis nebo zašifrovaní odeslána do karty, kde se provedou požadované informace a zpátky k uživateli již přichází výsledek. [8]

3.3.2 Správa a distribuce veřejných klíčů

Jak již bylo zmíněno PKI (Public Key Infrastructure) se zabývá správou a distribucí veřejných klíčů z asymetrické kryptografie. Díky tomuto komplexnímu systému organizačních, technických a legislativních nástrojů je možné používat elektronický podpis v praxi. Hlavními součástmi PKI je certifikační autorita a registrační autorita.

[10] Certifikát je elektronicky podepsaný veřejný šifrovací klíč, který vydává certifikační autorita.

- Certifikační autorita - smyslem této autority je vydávání certifikátů, jejich následná správa a poskytování certifikátů. Certifikační autorita ověřuje digitální podpisy a také zneplatňuje certifikáty, které nejsou již považovány za bezpečné. Dále také zveřejňuje seznamy nedůvěryhodných certifikátů.

- Registrační autorita - tato autorita ověřuje identity žadatele o vydávání certifikátů. Dá se využít i jako pomoc při zpracování žádosti o certifikát.

Systém PKI tedy reprezentuje věrohodné předání veřejného klíče od jeho vlastníka k jeho uživateli, ověřovateli digitálního podpisu. Aby k tomuto předání došlo je nutné důvěryhodným způsobem získat certifikát s veřejným klíčem dané certifikační autority. [8]

3.3.3 Šifrování

Zabezpečení dat před neoprávněným přístupem je řešitelné pomocí různých opatření. Mezi nejúčinnější patří vhodně a správně implementovaná kryptografická ochrana. Bezpečnost šifry nejvíce závisí na kvalitě použitého klíče. Je nutné, aby klíč byl dostatečně komplexní a náhodný. Pokud tak nenastane, šifra je snadno prolomitelná. Potřebné hlavní klíče se generují a ukládají v bezpečném systému pro správu klíčů a následně jsou importovány na čipovou kartu pro standardní použití. Programové vybavení a čipová karta umožňují šifrovat a dešifrovat pracovní šifrovací klíče, které se využívají k vlastnímu šifrování nebo dešifrování dat. Pokud nastane porucha nebo ztráta karty je možné hlavní klíče importovat na náhradní kartu. [5]

Čipové karty podporují symetrické a asymetrické šifrovací algoritmy. Symetrické šifrovací algoritmy využívají pouze jediného klíče, kdežto asymetrické dvojici veřejného a soukromého. Na čipových kartách jsou nejčastěji implementovány algoritmy DES, 3DES a AES pro symetrické šifrování a RSA a hashovaní funkce pro šifrování asymetrické.

- DES, 3DES - neboli Data Encryption Standart; tento blokový algoritmus[11] byl vyvinut v 70. letech. V dnešní době je považován DES za nespolehlivý s ohledem na velikost jeho klíče (64 bit) a také na to, že v roce 1997 byla tato šifra prolomena. V návaznosti na tento fakt vznikl algoritmus 3DES, který je provádí trojitou aplikaci DES. 3DES běžně pracuje s klíčem o celkové délce 168 bitů. Ačkoli 3DES je v porovnání s AES daleko pomalejší, stále se používá při šifrování SSL spojení.[12]

[11] Blokový algoritmus využivá pevně stanovený počet bitů = blok, který poté zašifruje.

[12] Protokol SSL (Secure Socket Layer) vytváří bezpečné šifrované propojení mezi uživatelem a serverem.

- AES - Advanced Encryption Standart - jedná se o standard, který byl udělen blokové šifře Rijndael. Vznikl jako náhrada za DES. Tato metoda může mít délku klíče 128, 192 nebo 256 bitů.

- RSA - tato šifra dostala název podle iniciály autorů Rivest, Shamir a Adleman. Tento algoritmus je vyhovující pro šifrování a zároveň pro podepisování díky tomu, že využívá rozkladu čísla na jeho prvočinitele. Algoritmus RSA se v současné době využívá nejvíce v systémech PGP[13]. [9]

- Hash funkce - jedná se o transformaci řetězce znaků o libovolné délce do řetězce znaků s pevnou délkou - otisk. Nejčastěji hash funkce slouží ke kontrole integrity dat, k rychlému porovnání dvojice zpráv nebo indexování. Hashovací algoritmus generuje otisk o určité délce. U MD5 je dlouhý 128 bitů (32 znaků) u SHA-1 160 bitů (40 znaků).

Hlavní výhodou symetrického šifrování je jeho rychlost, čehož se využívá pro šifrování a dešifrování velkého množství dat. Bohužel je ale použití jednoho sdíleného klíče velmi nepraktické s ohledem na bezpečnost. Oproti tomu asymetrické šifrování je bezpečnější, zato ale výrazně pomalé a prakticky nepoužitelné pro šifrování velkého množství dat. V praxi se velmi často obě metody šifrování kombinují tzv. vlastní data se zašifrují pomocí symetrického algoritmu (AES) a použitý klíč se zašifruje pomocí asymetrického algoritmu (RSA).

3.4 Technologie Microsoft for SmartCards

Společnost Microsoft již dlouhou dobu spolupracuje s výrobci čipových karet. Nejznámější spolupráce nastala se společností Gemalto (Společnost Gemalto vznikla spojením firem Axalto a Gemplus). Během této spolupráce se podařilo vyvinout platformu operačního systému Microsoft Windows také pro smartkarty.

Microsoft Windows for Smart Cards je operační systém založený na architektuře, která podporuje různé čipy a platformy. Je rozšiřitelný a podporuje rostoucí počet výrobců karet. Operační systém běží na Windows Powered Smart Card, což je mikropočítač bez grafického uživatelského rozhraní. Aplikace této technologie je možné vytvářet v Microsoft visual basic nebo v Microsoft Visual C.

Fyzické vlastnosti technologie jsou vestavěné do hardware Windows Powered Smart Card. Využívá především software pro ochranu oprávněného přístupu ACL (Access control list). Navíc tyto karty využívají MS-DOS typ souborový systém, který ukládá jednotlivé aplikace na různá místa v paměti. Díky ALC, souborovému

[13]PGP - Pretty Good Privacy, jedná se o počítačový program, který využívá RSA. Tento program se používá při šifrování emailů, PGP je součástí většiny e-mailových klientů.

systému, zabezpečeným knihovnám a matematickým algoritmům není schopen neautorizovaný uživatel využít invazivní manipulace. Tak jak to platí u platebních nebo bankovních karet, i Windows Powered Smart Card je možné při krádeži deaktivovat a aktivovat vydání nové karty. [11]

Pro ladění a vytváření aplikací je možné využít Microsoft Visual Studio, který bez problémů komunikuje s Windows Smart Powered Cards. Kromě toho je možné využít Microsoft Windows Smart Card Toolkit.

- Windows for Smart Cards - jedná se o logické rozšíření operačních systémů Windows, dále poskytuje soudržný rozvoj a testovací prostředí.
- Windows Smart Cards Toolkit - je možné vytvářet a ladit mnoho různých aplikací ve stejném čase pro různá prostředí.
- Windows Powered Smart Cards s operačním systémem Windows umožňuje ukládání osobních kontaktní informací.

3.4.1 Specifické požadavky

Komunikace mezi počítačem a čipovou kartou probíhá podle specifikace PC/SC (Personal computer/Smart Card). Jedná se o specifikaci implementovanou už od verze operačního systému Microsoft Windows 200x/XP. Dále pro fungování je nutné mít instalované tyto komponenty:

- Ověřování na základě certifikátu služby (Certificate Authentication Service) - pomocí této služby dokáže systém Windows vytvářet certifikáty, které lze instalovat na čipové kartě.
- Microsoft Active Directory - jedná se o kompatibilní adresářové služby, kde jsou uloženy certifikáty.
- Cryptographic Service Provider (CSP) - poskytovatel kryptografických služeb usnadňuje komunikaci mezi zařízením a čipovou kartou.
- Windows 2000 nebo Windows Xp - nejnižší verze systému windows, se kterými je možné pracovat. Tyto operační systémy využívají PC/SC kompatibilní čtečky čipových karet s ověřovacím protokolem Kerberos[14].

[14]Kerberous - jedná se o síťový autentizační protokol umožňující komunikaci v nezabezpečené síti. Tento protokol, který je postaven na symetrické kryptografii zamezuje odposlechu a zaručuje integritu.

3.4.2 Zajištění bezpečnosti

Technologie Microsoft for Smard Card využívá bezpečnostních algoritmů (RSA, DES, 3DES, AES a SHA), podporuje také digitální podpis a i více autentizačních mechanismů jako jsou PIN, otisk prstu nebo oční sítnice.

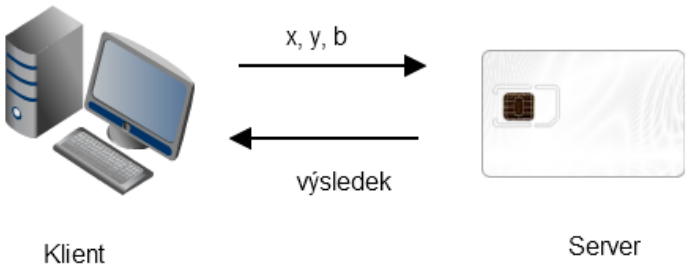
Autentizaci klienta zahrnuje zjišťování a jeho ověřování na server pro vytvoření bezpečného komunikačního kanálu. Zabezpečené protokoly, Secure Sockets Layer (SSL) nebo Transport Layer Security (TLS), se obvykle používají ve spojení s důvěryhodným veřejným klíčem certifikátu poskytnutých klientem, který identifikuje klienta na serveru. Klient může mít spuštěn Internet Explorer na platformě Windows, a server může být IIS (Internet Information Service), nebo nějaký jiný webový server, který podporuje SSL/TLS.

Zabezpečené relace jsou stanoveny pomocí veřejného klíče a výměnou unikátního klíče relace, který pak může být použit k zajištění integrity dat a důvěrnosti celé relace. Je možné dosáhnout dodatečného ověření mapování certifikátu na účet uživatele nebo skupiny s dříve stanoveným přístup-kontrola oprávnění. Čipové karty zvyšují ověření pomocí veřejného klíče tím, že slouží jako bezpečné úložiště pro klíč soukromý. [11]

4 NÁVRH PROJEKTU A JEHO REALIZACE

Tato práce se zabývá jak návrhem optimalizace výpočtů, tak i jejich realizací na čipové kartě. Jak již z teorie vyplynulo, v rámci optimalizace je nasnadě vycházet z modulární aritmetiky a jejich pravidel. Na čipovou kartu jsou implementovány dva algoritmy, jeden pro modulární sčítání a druhý pro modulární násobení.

Komunikační schéma mezi čipovou kartou a počítačem funguje na principu klient-server. V tomto případě klienta symbolizuje počítač, který je přímo závislý na čipové kartě - serveru. Klient zasílá serveru pouze hodnoty, veškeré výpočty se odehrávají na straně čipové karty. Viz obr. B.13.



Obrázek 4.1: Zjednodušené schéma komunikace.

4.1 Návrh matematických operací

Při návrhu matematických operací bylo nezbytné zjistit fyzické parametry karty. Mezi rozhodující patřily především informace o typu procesoru, s jakými daty procesor(karta) pracuje a i velikost dostupné paměti. Čipová karta, na které jsou implementovány výpočty disponuje 32 bitovým procesorem, velikost dostupné paměti se pohybuje kolem 40 KB (tato hodnota se bude snižovat s ohledem na knihovny a programy, které je nutné pro spuštění výpočtů na kartu nahrát). Další zajímavostí bylo zjištění, že čipová karta podporuje pouze datový formát *ulong*, což představuje 2^6bitů tj. 64 bitové číslo.

Pro matematické operace - sčítání i násobení je využito možnosti zapsat velké číslo (o několika řádech) ve tvaru $s = (a_n a_{n-1}..a_1 a_0)_b$ kde $b \in N/1; a_i \in 0..b-1$. V tomto případě b symbolizuje základ, reprezentaci čísla. Pokud jsou dvě velká čísla takto zapsaná při stejném b (základní podmínka, bez ni by se tento způsob nedal aplikovat) lze počítat s jednotlivými řády samostatně - viz operací sčítání, násobení. [19]

Sčítání

V rámci sčítání velkých čísel je nutné brát ohled na velikost(délku) čísel se kterými provádíme danou operaci. Bohužel čísla v řádu několika stovek jsou mnohem náročnější na zpracování než-li čísla v řádech desítek. Pro následnou implementaci výpočtu na čipovou kartě byl zvolen řád 6 s ohledem na fakt, že výpočty pro kartu jsou hardwarově omezeny. Při návrhu operace sčítání se počítalo s čísly o řádu 8 viz níže tab. 4.1. Operace součtu probíhá následovně:

1. Jsou k dispozici čísla x a y, které mají velikost řádu u a v, čísla x a y mají stejnou reprezentaci b. Výpočty probíhaly s hodnotou $b = 10$. Při implementaci bude nutné znovu zvážit, zda je stále možnost používat základ 10, zda čipová karta dokáže s tímto základem počítat.

2. Dále je definováno c, které pomáhá při výpočtu součtu. Hodnota c obsahuje podmínku: $w_i = (x_i + y_i + c) < b$; $c = 0$; pokud tato podmínka neplatí $c = 1$.

3. Následuje určení řádu vypočítaného čísla i, podmínka je definována jako $u > v$ $i = u$; pokud podmínka neplatí $i = v$.

4. Probíhá cyklus operace sčítání $w_i = (x_i + y_i + c) mod b$.

i	8	7	6	5	4	3	2	1	0
x_i	1	2	3	4	5	6	7	8	9
y_i	1	1	2	2	3	3	4	4	5
w_i	2	3	5	6	9	0	2	3	4
c	0	0	0	0	0	1	1	1	1

Tabulka 4.1: Princip sčítání pro čísla x = 123456789 a y=112233445.

Způsob sčítání jde aplikovat i pro čísla s vyššími řády. Chtějme získat součet dvou libovolných celých kladných čísel o řádu 35 tab. 4.2. V tomto případě je možné rozdělit výpočet na několik částí, ve kterých proběhne výše popsaný algoritmus a poté je opět seskupit v daném pořadí (dle čísla řádu) za sebe. Zpětné seskupení v sobě skrývá také nutnost ošetřit hodnotu c, pokud by se tak nestalo může dojít ke znehodnocení celého algoritmu a ztrátě dat.

Násobení

Násobení velkých čísel je považováno za náročnější operaci než samotné sčítání velkých čísel. V rámci násobení se postupuje obdobně jako u sčítání, výpočty probíhají po znacích, metoda výpočtu je známá jako "metoda tužky a papíru". Tuto operaci jde dále zefektivnit pomocí Montgomeryho redukce, která při svém výpočtu používá jak modulárního násobení tak i hledání největšího společného násobku.

i	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x_i	4	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
y_i	5	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7	7
w_i	0	0	0	0	0	0	2	2	2	2	2	2	4	4	4	4	4	3
c	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
i	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18
x_i	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
y_i	2	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4
w_i	3	3	3	3	3	3	5	5	5	5	5	5	7	7	7	7	7	8
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Tabulka 4.2: Součet čísel o řádu 35.

Operace násobení probíhá následovně:

1. Jsou k dispozici čísla x a y, které mají velikost řádu u a v, čísla x a y mají stejnou reprezentaci b. Výpočty probíhaly s hodnotou $b = 10$. Při implementaci bude nutné znovu zvážit, zda je stále možnost používat základ 10, zda čipová karta dokáže s tímto základem počítat.

2. Dále je definováno c, které pomáhá při výpočtu součinu. Standardně je nastaveno na nulu, v pozdějších krocích slouží pro uložení mezivýsledků.

3. Výpočet součinu je složen z následujících kroků, které jsou definovány v podobě cyklů:

   (a) Dojde k určení řádu vypočítaného čísla pomocí for cyklu, který se opakuje od 0 do $u + v + 1$.

   (b) Následuje určení prvního čitatele, který se nachází v nultém řádu čísla y.

   (c) Výpočet proběhne jako $w_{i+j} = (x_j * \mathrm{y}_i + c)$.

4. Výsledek je poté vložen do pomocných proměnných k a l, kde k představuje nultý řád w a l řád první, hodnota k bude poté vložena do c a celý cyklus se opakuje.

5. V poslední fázi se čísla uložené v jednotlivých krocích ve w_{i+j} sečtou a dají konečný výsledek. Viz. praktická ukázka výpočtu tab. 4.3.

i	j	c	$w_{i+j} + x_j y_i + c$	k	l	w_6	w_5	w_4	w_3	w_2	w_1	w_0
0	0	0	$0 + 24 + 0$	2	4	0	0	0	0	0	0	4
	1	2	$0 + 18 + 2$	2	0	0	0	0	0	0	0	4
	2	2	$0 + 12 + 2$	1	4	0	0	0	0	4	0	4
	3	1	$0 + 6 + 1$	0	7	0	0	0	7	4	0	4
1	0	0	$0 + 20 + 0$	2	0	0	0	0	7	4	0	4
	1	2	$4 + 15 + 2$	2	1	0	0	0	7	1	0	4
	2	2	$7 + 10 + 2$	1	9	0	0	0	9	1	0	4
	3	1	$0 + 5 + 1$	0	6	0	0	6	9	1	0	4
2	0	0	$1 + 16 + 0$	1	7	0	0	6	9	7	0	4
	1	1	$9 + 12 + 1$	2	2	0	0	6	2	7	0	4
	2	2	$6 + 8 + 2$	1	6	0	0	6	2	7	0	4
	3	1	$0 + 4 + 1$	0	5	0	5	6	2	7	0	4

Tabulka 4.3: Princip násobení pro čísla 1234 a 456.

4.2 Simulace matematických operací

Správnost výše uvedených matematických operací byla ověřena simulací v prostředí Microsof Visual Studio 2008 a pomocí programovacího jazyka C#. Simulace spočívala ve zjištění času, který je potřeba pro provedení jednoduchého výpočtu. Pro simulaci byla použita velká čísla (v řádech 30 až 160 cifer) viz obr. 4.2 . Výsledky simulace jsou zobrazeny v tab. 4.4a 4.5. Podrobnosti k jednotlivým měřením lze nalézt v příloze stejně tak jako graf vyjadřující časovou závislost na počtu řádu u čísel. B.1

Nasimulování chování algoritmů probíhalo na notebooku značky ASUS - A6T, fyzické parametry, procesor AMD Turion TL52 1,61 GHz a 1,00 GB paměti RAM. I přes daleko lepší hardwarové vybavení notebooku byly simulované algoritmy navrženy tak aby fungovaly i na "hardwarově slabších" smartkartách.

Největší úskalí v rámci programování bylo vhodně zvolit datový typ, který dokáže zpracovat velké množství čísel. Řešením se ukázalo použít textový typ `String`, kde se čísla zapisují jako znaky. Poté už v rámci operací a podmínek, lze k těmto číslům přistupovat a ze `String` brát jednotlivě znaky a ukládat je do bloků o určité délce nebo je konvertovat do jiných datových typů - `Int`.

Simulační program [1] byl vytvořen v programovacím jazyku C# ve kterém probíhá i vývoj aplikací pro smartkarty ve frameworku. V hlavním vláknu programu `Program` ve funkci `Main` jsou volány jednotlivé funkce třídy `Knihovna` a také se zde

[1] Kompletní zdrojový kód simulačního programu je přiložen na CD

měří čas potřebný pro simulované výpočty.

Třída `Knihovna` obsahuje vlastní výpočty simulačního programu. Jednotlivé výpočty jsou volány pomocí následujících funkcí:

- `soucetPoCislech` - součet po jednotlivých číslicích
- `soucetPoBlocich` - součet po blocích číslic
- `nasobeni` - násobení

Při použití jedné z funkcí jsou vždy předávány jako vstupní parametry dvě čísla, které chceme sečíst nebo mezi sebou vynásobit.

Funkce pro sčítání nejprve pomocí funkce `rozdeleniCiselDoBloku` vstupní data rozdělí do bloků, počet číslic v jednom bloku je definován při vytváření nového objektu třídy `Knihovna` jeho konstruktorem. V dalším kroku je v cyklu opakovaně volána funkce pro součet jednotlivých bloků a v tomto cyklu také probíhá kontrola na případné přetečení při součtu dvou bloků. Pro sčítání po jednotlivých číslicích se v cyklu volá funkce `soucetJedenBlokPoCislech`, pro sčítání po blocích pak `soucetJedenBlokPoBlocich`.

Operace násobení je prováděna přímo ve funkci `nasobeni`, ve které neprobíhá rozdělování čísel do bloků ale je přímo aplikován způsob násobení popsaný v kap. Návrh matematických operací.

velikost čísla x (počet řádů)	30	60	90	120	150	180
velikost čísla y (počet řádů)	30	60	90	120	150	180
	$[\mu s]$	$[\mu s]$	$[\mu s]$	$[\mu s]$	$[\mu s]$	$[\mu s]$
sčítání-jednotlivá čísla	2,9794	3,0191	3,0736	3,1186	3,2020	3,2381
sčítání-bloky	1,0850	1,1053	1,1359	1,3484	1,1742	2,7956
násobení	1,1188	1,4453	2,4053	2,4650	2,7179	3,9176

Tabulka 4.4: Simulace výpočtů pro čísla se stejnou délkou řádů.

Tabulky 4.4 a 4.5 zobrazují jednotlivé časy pro tři druhy výpočtů.

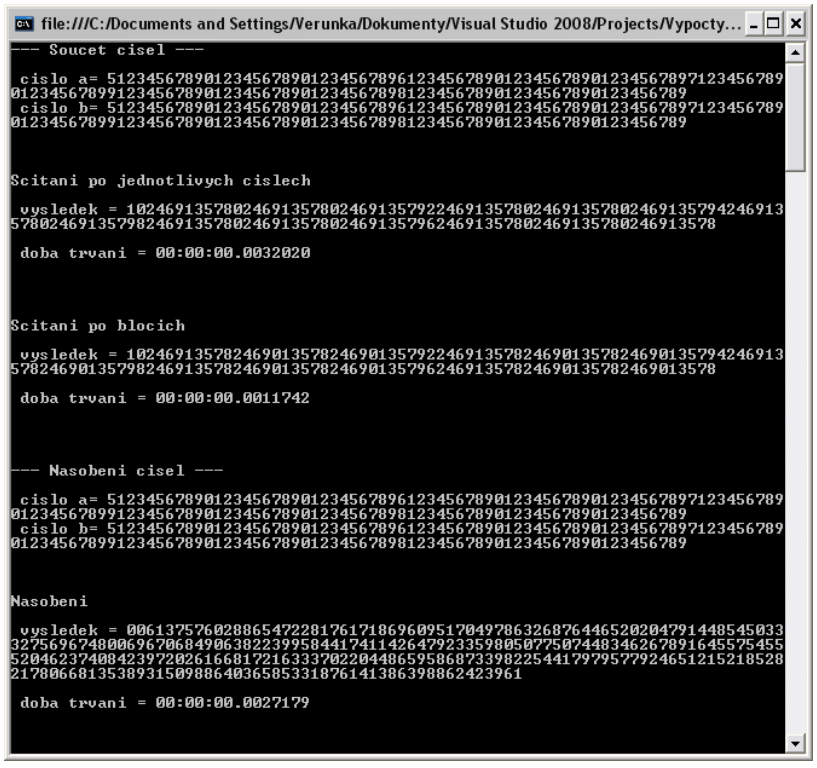
- Sčítání - jednotlivá čísla - princip spočívá v sečítání jednotlivých řádů čísel, postupuje se od nultého řádu, směrem k nejvyššímu, ukázka postupu viz tab. 4.1
- Sčítání - bloky - tento algoritmus vychází z podstaty rozdělení velkých čísel do bloků o určité délce, po rozdělení nastane součet čísel, kdy se bloky sčítají postupně.4.2

velikost čísla x (počet řádů)	30	60	90	120	150	180
velikost čísla y (počet řádů)	10	30	30	30	30	30
	$[\mu s]$	$[\mu s]$	$[\mu s]$	$[\mu s]$	$[\mu s]$	$[\mu s]$
sčítání-jednotlivá čísla	2,9716	3,0207	3,0627	3,8520	3,1448	3,2342
sčítání-bloky	1,0675	1,0904	1,0951	1,1824	1,1194	1,1870
násobení	1,0415	1,2259	2,1352	2,2048	2,3049	1,4699

Tabulka 4.5: Simulace výpočtů pro čísla s různou délkou řádů.

- Násobení - pracuje na principu násobení pomocí tužky a papíru. 4.3

Z naměřených hodnot vyplývá, že čím je číslo nebo čísla, která je nutné zpracovat, větší tím zákonitě se navyšuje doba na jejich zpracování. Zajímavým poznatkem je konstantní hodnota času při sčítání čísel s různou délkou řádů pomocí blokovou metodou, naopak tato technika při sčítání jednotlivých řádů vykazuje lineární průběh.



Obrázek 4.2: Ukázka simulace výpočtů pro čísla se stejnou délkou řádů.

4.3 Realizace

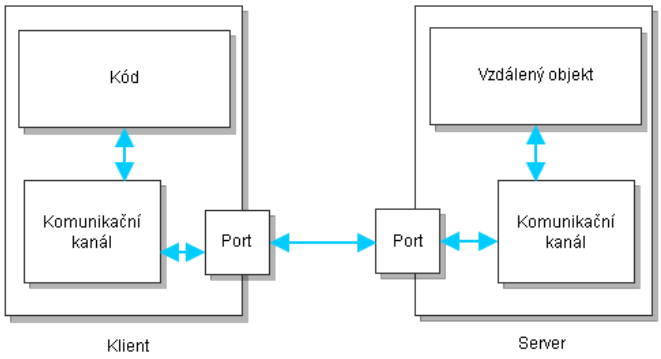
Popis prostředí a komunikačního modelu

Pro realizaci projektu byla zvolena technologie .NET Card, jejíž součástí není pouze programovatelná smartkarta vytvořená pro práci s .NET aplikacemi, ale i .NET SmartCard Framework obsahující důležité knihovny, které vyžadují aplikace běžící na smartkartách. Součástí technologie .NET Card jsou i nástroje pro vývoj nových aplikací uzpůsobených na Gemalto .NET cards a Common language runtime (CLR). Framework .NET Smart Card má v sobě implementovaný model komunikace typu klient-server, ve kterém terminál nebo počítač představují klienta a Gemalto .NET Card server. Interakce mezi těmito dvě subjekty zahajuje vždy klient použitím request nebo reply protokolu. Jako transportní protokoly se používají TCP nebo HTTP dle standardu ISO 7816-4. Veškerá komunikace mezi klientem a serverem prochází tzv. kanálem. Framework .NET Smart Card definuje vlastni kanál - `APDUChannel`, který je zodpovědný za kódování metod do binárního formátu a následný přenos dat z klienta na server. [18]

Aby tato komunikace byla realizovatelná, je nutno zaregistrovat kanál a číslo portu po kterých bude komunikace probíhat. (Viz obr. 4.3). Jak na klientské tak i na serverově straně se inicializuje nový kanál s daným portem, klientská strana navíc použije třídu `Activator` pro získání odkazu na vzdálený objekt, který je definován URL adresu a název vzdálené třídy poté obdrží hodnotu URI. URL poté vypadá následovně:

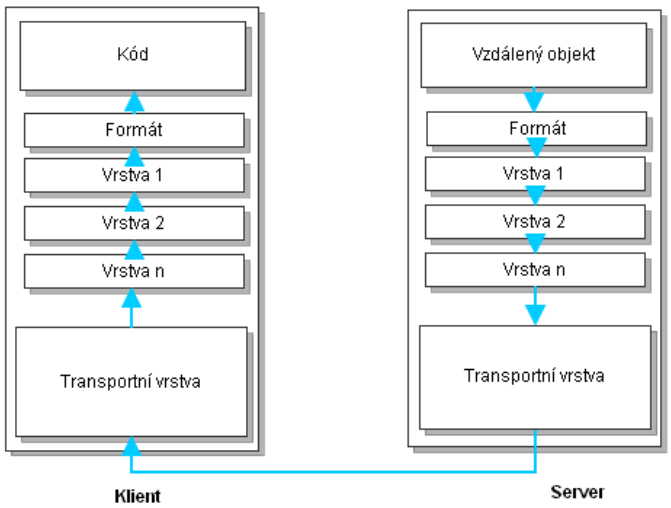
`apdu://název čteciho zařízení:port registrované služby/název služby`

Mimo přesného názvu čtecího zařízení lze uvést parametr `promptDialog` (umožňuje uživateli vybrat si čtecí zařízení dle vlastního výběru) `selfDiscover` (automaticky je vybráno čtecí zařízení, ve kterém se nachází smartkarta).



Obrázek 4.3: Komunikační schéma v prostředí .NET Smart Card [18].

Čipové karty v rámci zjednodušení komunikace využívají vzdáleného přístupu k aplikacím. Respektive klient i server pracuji v několika vrstvách, které umožňují vlastní modifikaci (šifrování, kompresi apod.) toku dat. Princip zobrazen na obr. 4.4.



Obrázek 4.4: Komunikační schéma pomocí vrstev v prostředí .NET Smart Card [18].

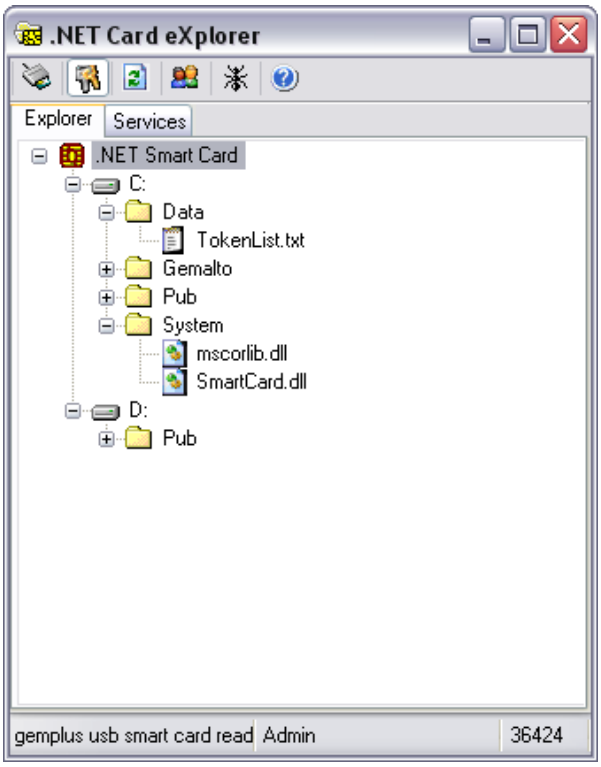
4.3.1 Práce se smartkartou

Během celého projektu byla využita smartkarta typu Gemalto.NET v2+. Tato smartkarta disponuje 32-bitovým mikroprocesorem značky Infenion a maximální přenosovou rychlostí 223200 bps. Dále obsahuje kryptografický koprocesor umožňující výpočty kryptografických operací jako jsou DES, TripleDES, Rijindael nebo RSA. Další technické parametry karty jsou k dispozici v příloze. A.1

Díky .NET Smart Card Frameworku a aplikaci .NET Card eXploreru lze zobrazit celý souborový systém smartkarty obr. 4.5. Ten je logicky rozdělen na dva disky C:\ a D:\. Disk C:\ obsahuje knihovny důležité pro zprovoznění aplikací na kartě (např. `SmartCard.dll`), na disku D:\ se nachází informace o fyzických parametrech a schopnostech karty. Datový prostor pro samotné aplikace je k dispozici na disku C:\ ve složce Public. [18]

Aplikace .NET Card eXploreru umožňuje práci s daty na kartě v podobě vytváření nových složek, nahrávání aplikací nebo jejich administraci v rámci souborového systému. Celková manipulace s daty je závislá na přístupové roli, veškeré modifikace může provádět pouze Administrátor, User nebo role Everyone mají velmi omezené možnosti úprav.

Karta obecně dokáže pouze pracovat s aplikacemi (s příponami .exe) a knihovnami (s příponami .dll), které jsou nahrány na kartu pomocí čtečky, pro tento projekt



Obrázek 4.5: Aplikace .NET Card eXplorer.

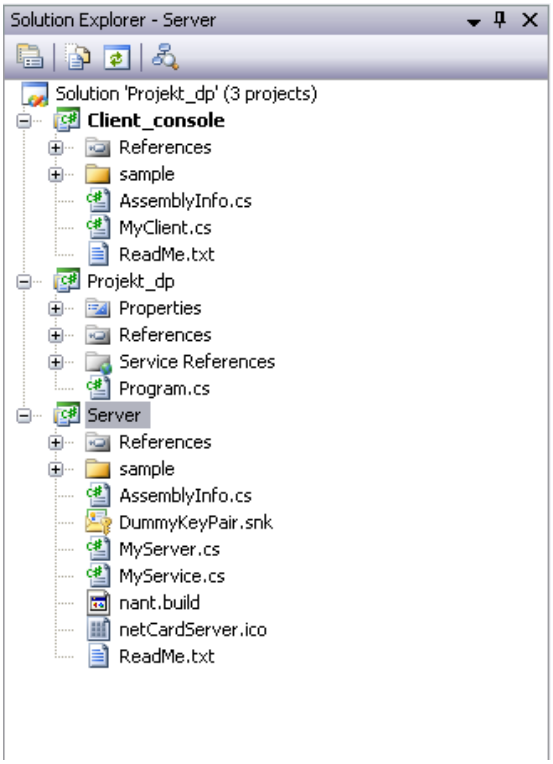
byla využita čtečka karet Gemplus USB Smart Card Reader 0. Nahrávání aplikací a knihoven umožňuje Card Explorer tool, kde je důležité vlastní osobu nejdříve autentizovat pomoci PIN. Po úspěšné autentizaci se zobrazí celý souborový systém a pro nahraní již stačí pouze kliknout na složku, ke které byl umožněn přístup, a přes pravé tlačítko myši vybrat Add > New File, poté vyhledat daný soubor určený pro nahrání > Open. Daný soubor se zkopíruje do vybrané složky na kartě. Spuštění souboru je poté spuštěn > Execute. Aplikace běžící na kartě jsou trvalé tzv. nedochází k jejich ukončení i např. při odebrání karty z čtecího zařízení. Jedinou výjimku tvoří nezaregistrovaní služby.

Gemalto .NET Card umožňuje řešit i zabezpečení dat, nejčastěji pomocí:

- Access Manager - umožňuje vytvářet uživatelské účty s různými pravomocemi. Tyto uživatelské účty "user roles" jsou převážně situovány pro práci s aplikacemi na kartě (v jejím souborovém systému).
- Application Security - aplikace, které jsou nasazeny na kartu, jsou vždy podepsané pomocí veřejného klíče.
- Data Security - data, která uložená na kartě jsou k dispozici pouze vybraným uživatelům pomocí "user roles".

4.3.2 Rozbor aplikace

Vývoj aplikace probíhal v Microsoft Visual Studio 2008 pomocí .NET Frameworku 3.5. Jako programovací jazyk byl zvolen C#. Veškeré ladění probíhalo v operačním systému Microsoft Windows XP, za pomoci vývojového prostředí pro smartkarty - SDK. Vytvořená aplikace se skládá ze dvou projektů jeden - Client_Console definuje procesy a funkce, které probíhají na straně klienta. Druhý projekt - Server popisuje to stejné jen na straně serveru.



Obrázek 4.6: Vytvořené projekty v MS Visual Studio 2008.

Projekt Server

Server aplikaci lze vytvořit pomocí šablony netCard Server, která je součástí vývojového prostředí Microsoft Visual Studiu 2008 .NET Framework 3.5. Díky použití šablony netCard Server se vygenerují i další soubory, především DummyKeyPair.snk, MyServer.cs, MyServices.cs a nant.build.

- `Server.csproj` - jedná se o hlavní soubor aplikace Server.
- `MyServices.cs` - třída, která spravuje služby, které jsou nabízeny klientům.
- `AssemblyInfo.cs` - třída, která slouží k aktivaci podpisu aplikací.
- `DummyKeyPair.snk` - je určen k podepsání souboru. Veškeré aplikace, které se přidávají na smartkartu musí být podepsané.
- `MyServer.cs` - hlavní třída, která obsahuje zdrojový kód obsluhující dění na serveru, také umožňuje registraci a zpřístupnění služeb.
- `nant.build` - základní soubor pro spuštění aplikace na čipové kartě.

Projekt Client_Console

Pro klientskou aplikaci lze využit také šablonu stejně jako pro Server aplikaci. V případě klientské aplikace se jedna o netCard Client Console. Po použití této šablony se vytvoří i další soubory důležité pro fungování, jedná se o:

- `Client_console.csproj` - hlavní soubor aplikace Client Console.
- `MyClient.cs` - hlavní třída umožňující přístup na služby, zveřejněné na serveru.
- `AssemblyInfo.cs` - obsahuje všeobecné informace o atributech definovaných v Client_Console.

Projekt_dp Aplikace

Aplikace s názvem Projekt_dp slouží ke kompilaci všech projektů, obsahuje mimo potřebných knihoven pouze třídu `Program.cs`, která zajišťuje chod navržené aplikace.

Implementace nasimulovaných algoritmů

V rámci implementace, měly být odsimulované matematické algoritmy nahrány na smartkartu, do aplikace Server. Přesněji měly být definovány jako služby, které by Server poskytoval svým klientům. Princip komunikace poté měl být následující:

1. Server zpřístupní své služby pro klienty, dojde i k zaregistrování kanálu pro přístup k této službě.

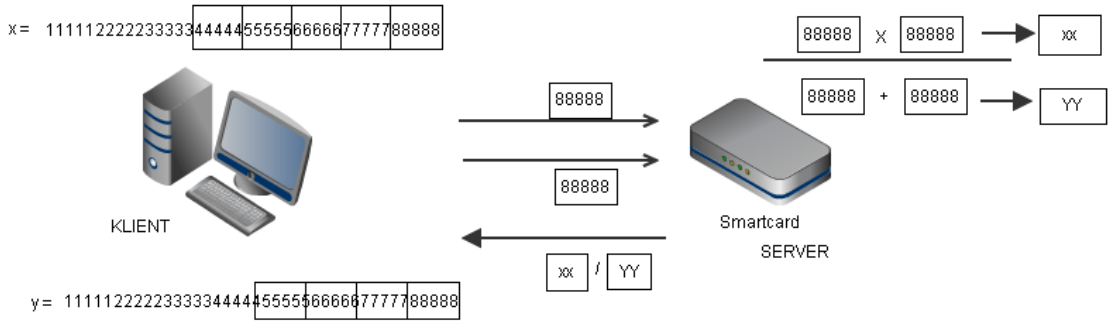
2. Klient je nucen se autentizovat pro přihlášení k Serveru.

3. Klient zaregistruje kanál a i samotnou službu, kterou chce využívat.

4. Klient zasílá službě provozované na Serveru vstupní informace.

5. Server zpracovává požadavek od klienta a zasílá mu výslednou odpověď.

Než začne samotné zasílání vstupních informací do služeb provozovaných na Serveru je nutné na straně klienta připravit vstupní informace do takové podoby, aby je Server mohl bez problémů přijmout a pracovat s nimi. Proto se rozdělí čísla rozdělí na malé bloky o velikosti n, které se postupně odesílají na Server, tam zvolená služba zpracuje a výsledek odešle zpátky klientovi. Poté se zpracuje další blok. Viz obr. 4.7



Obrázek 4.7: Průběh zpracování vstupních dat.

Nasimulované algoritmy nebyly implementovány na smartkartu z důvodu nemožnosti nahrát na kartu jakýkoliv kód. Nahrání a následnému ověření funkčnosti bylo zrušeno díky aplikaci .NET Card eXploreru, který neumožnil uploadovat na kartu debugovaný soubor `Server.exe`. Tento problém, mohl vzniknout již plnou pamětí smartkarty, bohužel i přes snahu o uvolnění místa na kartě se nepodařilo soubor uploadovat. Jediná možnost, je zkusit přehrát defaultní nastavení smartkarty a pokusit se nahrát soubor znova.

5 ZÁVĚR

V diplomové práci jsem se zaměřila na problematiku bezpečnostních algoritmů a na jejich optimalizaci. V rámci teoretického úvodu byly připomenuty základní poznatky z algebry, přesněji týkající se modulární aritmetiky, která tvoří základ pro fungování každého bezpečnostního algoritmu. Dále byla zmíněna důležitost reprezentace velkých čísel a principy prvočíselnosti. Součástí matematicky zaměřeného úvodu byly zmíněny ukázky šifer využívající modulární aritmetiku.

Mou snahou bylo také v této práci rozebrat teorii čipových karet, principy jejich fungování, základní rozdělení a zabezpečení dat. Poté byla v rámci čipových karet zmíněna i platforma .NET Smart Card framework umožňující vytváření aplikací pro čipové karty a operačním systému určených pro čipové karty firmy Microsoft - Microsoft for Smart Card.

Kapitola Návrh projektu a jeho realizace popisuje návrh změn, které jsou podloženy teoretickými důkazy. Tato práce se zaměřila na optimalizaci výpočtu součtu a součinu dvou velkých čísel (řádově desítek a stovek). Návrhy byly realizovány ve vývojovém prostředí Microsoft Visual Studio 2008 v jazyku C#. Výsledky simulace jsou k dispozici v tabulkách 4.4 a 4.5. Graf závislosti je také součástí přílohy.

Samotné nasazené nasimulovaných výpočtů se bohužel nepodařilo implementovat na smartkartu, důvod neúspěchu samotné implementace může být v hardwarovém omezení smartkarty (omezená velikost paměti) nebo ve špatném nahrání spustitelného .exe souboru na server. Na základě teoretických znalostí poznatků v této diplomové práci a provedené simulaci výpočtu by měly být navržené algoritmy funkční pro implementaci do čipových karet.

REFERENCE

[1] BAY, R. Čipové karty a USB tokeny, aneb bezpečnější autentizace a šifrování (3) - obecné požadavky na tokeny. Svět sítí [online] 2003 [cit. 2010-11-17]. Dostupný z URL: <http://www.svetsiti.cz/print.asp?rubrika=Tutorialy&clanekID=267/>

[2] BAY, R. Čipové karty a USB tokeny, aneb bezpečnější autentizace a šifrování (4) - bezpečnostní požadavky na tokeny I. Svět sítí [online] 2003 [cit. 2010-11-18]. Dostupný z URL: <http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=264&clanekID=268>

[3] BAY, R. Čipové karty a USB tokeny, aneb bezpečnější autentizace a šifrování (5) - bezpečnostní požadavky na tokeny II. Svět sítí [online] 2003 [cit. 2010-11-18]. Dostupný z URL: <http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=264&clanekID=269>

[4] KADLČÁKOVÁ L. Čipové karty, PKCS#11, PKCS#15 Aplikovaná kryptografie [online] 2006 [cit. 2010-11-17]. Dostupný z URL: <http://www.karlin.mff.cuni.cz/ kadlcak/lessons/ak.html>

[5] ROSOL I. Čipové karty IT Security [online] 2010 [cit. 2010-11-17]. Dostupný z URL: <http://www.systemonline.cz/it-security/cipove-karty.htm>

[6] KOŽUŠNÍK D. Potřebuje elektronický podpis bezpečí čipových karet? [online] 2001 [cit. 2010-11-18]. Dostupný z URL: <http://www.systemonline.cz/clanky/potrebuje-elektronicky-podpis-bezpeci-cipovych-karet.htm>

[7] DOLEŽAL D. Jak funguje digitální podpis [online] 2002 [cit. 2010-11-18]. Dostupný z URL: <http://http://interval.cz/clanky/jak-funguje-digitalni-podpis/>

[8] ŠABATOVÁ I. Elektronický podpis – principy a použití [online] 2005 [cit. 2010-11-18]. Dostupný z URL: <http://www.sabi.cz/stranky/FRAME_publikace.html>

[9] RYDVAL S. Algoritmus RSA [online] 2005 [cit. 2010-11-20]. Dostupný z URL: <http://www.rydval.cz/phprs/view.php?cisloclanku=2005123124>

[10] CHEN Z., DiGiorgio R. Understanding Java Card 2.0 [online] 1998 [cit. 2010-11-24]. Dostupný z URL: <http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html?page=1>

[11] DE CLERCQ J. Smart Cards [online] 2010 [cit. 2010-11-29]. Dostupný z URL: <http://technet.microsoft.com/en-us/library/dd277362.aspx>.

[12] KUBANDA, S. Modulárna aritmetika pre kryptografické výpočty v čipových kartách. Brno, 2010. 58 s. Diplomová práce. Masarykova univerzita, Fakulta informatiky.

[13] PŘIKRYL, J.; VLČEK, M. Modulární aritmetika, Malá Fermatova věta, Čínská věta o zbytcích. Matematické algoritmy, 2010. s. 38. Dostupné z URL:< http://euler.fd.cvut.cz/predmety/ma/files/ma-04-2008.pdf>.

[14] PŘIKRYL, J.; VLČEK, M. Prvočísla a dělitesnost. Matematické algoritmy, 2010. s. 40. Dostupné z URL:< http://euler.fd.cvut.cz/predmety/ma/files/ma-03-2008.pdf>.

[15] JURAS, S. Autentizace pomocí smartkaret. Brno, 2010. 68 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[16] BURDA, K. Bezpečnost informačních systémů. Brno, 2005. 104 s. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[17] LÓRENCZ, R. Aplikovaná numerická matematika a kryptografie. Praha, 2004. 97 s. České vysoké učení technické v Praze, Fakulta elektrotechnická.

[18] Gemalto .NET v2/2+ Smart Card [online].[2008], [cit. 2011-05-17].Dostupný z WWW:< http://www.gemalto.com/dwnld/5763Gemalto.NETUserGuide.pdf>.

[19] MENEZES, Alfred; OORSCHOT, Paul; VANSTONE, Scott. Handbook of Applied Cryptography. [s.l.]: CRC Press, 1996. Efficient Implementation, s. 591-634. ISBN 0-8493-8523-7.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ACL Seznam oprávnění připojený k objektu. – Access control list

AES Standard pro symetrické šifry. – Advanced Encryption Standart

AID Číslo, které je přidělené každému java appletu. – Application Identifier

APDU Formát zprávy, který zajišťuje komunikaci mezi čtecím zařízením a čipovou kartou. – Aplication Protocol Data Unit

API Programovatelné rozhraní aplikace. – Application Interface

CPS Jedná se o knihovnu v systému Microsoft Windows, která umožňuje kódování a dekódování funkcí. – Cryptographic Service Provider

CPU Procesor - součást počítače, která vykonává spuštěné programy. – Central Processing Unit

DES Symetrická šifra používající klíč o délce 64 bitů. – Data Encryption Standart

DF Adresář aplikace v Souborovém systému - týká se organizace dat v paměti. – Dedicated File

DSA Standard pro digitální podpis. – Digital singature algorithm

RAM Typ elektronické paměti, jedná se o kombinaci pamětí ROM a RAM. – Electrically Erasable Programmable Read-only Memory

EF Datový soubor v Souborovém systému - týká se organizace dat v paměti. – Elementary File

ECC Kryptografie eliptických křivek - založená na problému diskrétního algoritmu. – Elliptic curve cryptography

JVM Sada počítačových programů, které využívají modulu virtuálního stroje ke spuštění dalších programů a skriptů vytvořených v jazyce Java. – Java Virtual Machine

MD5 Hašovací funkce, která vytváří ze vstupních dat otisk o fixní délce. – Message-Digest algorithm

MF Kořenový adresář v Souborovém systému - týká se organizace dat v paměti. – Master File

MS-DOS Operační systém určený pro jednoduchou obsluhu. – Microsoft Disk Operating System

OS Operační systém je programové vybavení počítače (software). – Operating system

PIN Osobní identifikační číslo, které slouží pro autentizaci uživatele. – Personal Identification Number

PKCS Jedná se o skupinu standardů vztahujících se pro kryptografii s veřejným klíčem publikovanou RSA Security.

PKCS#11 Standard pro rozhraní kryptografický token - bezpečnost založena na hardware.

PKI V asymetrické kryptografii se tak označuje infrastruktura správy a distribuce veřejných klíčů. – Public Key Infrastructure

PGP Počítačový program, který využívá RSA šifrování. – Pretty Good Privacy

RAM Typ elektronické paměti, která umožňuje zápis a editaci dat. – Random-access Memory

ROM Typ elektronické paměti, na které nejdou editovat již zapsané informace. – Read-Only Memory

RSA Šifra s veřejným klíčem. – Rivest-Shamir-Adleman cipher

SHA-1 Rozšířená hašovací funkce, která vytváří ze vstupních dat otisk fixní délky. – Secure Hash Algorithm

SIM Identifikační účastnická karta se zabudovaným mikročipem, která souží pro identifikaci v mobilních sítí – Subscriber identity module

SSL Protokol, který poskytuje zabezpečení komunikace šifrováním a autentizací komunikujících stran. – Secure Sockets Layer

TLS Protokol, který poskytuje zabezpečení komunikace šifrováním a autentizací komunikujících stran. – Transport Layer Security

URI Identifikátor zdroje, zda se jedná o dokument či službu – Uniform Resource Identifier

URL Řetězec znaků s definovanou strukturou, který slouží k označení umístění zdrojů informace – Uniform Resource Locator

USB Univerzální sériová sběrnice, používá se pro připojení periférií k počítači, nástupce portů PS/2 – Universal Serial Bus

3DES Symetrická šifra používající klíč o délce 168 bitů. – Triple Data Encryption Standart

SEZNAM PŘÍLOH

A PRVNÍ PŘÍLOHA

A.1 Fyzické parametry Gemalto.NET v2+

- product

  name: Gemalto.NET v2+; vendor: Gemalto

  OSversion: 2.1.213.9175; CLRversion: 2.1.213.9175

- hardware

  name: SLE88CFX4000P

  version: B17; vendor: Infineon

  serialnumber: 57011351283EDE0B2F15FFFF

- comm speed

  9600; 19200; 38400; 55800; 76800; 111600; 115200; 223200

- crypto

  DES - type:sym, minimal key length:64, maximal key length:64, keygen

  TripleDES - type:sym, minimal key length:128, maximal key length:192, keygen

  Rijndeal - type:sym, minimal key length:128, maximal key length:256, keygen

  RSA - type:asym, minimal key length:256, maximal key length:2048, keygen

  MD5 - type:hash

  SHA1 - type:hash

  SHA256 - type:hash

  HMACSHA1 - type:hash

- converter - version: 3.2

- manufacturing

  siteID: 1982; date: 8340

  embedding:

  siteID: 0443; date: 8340

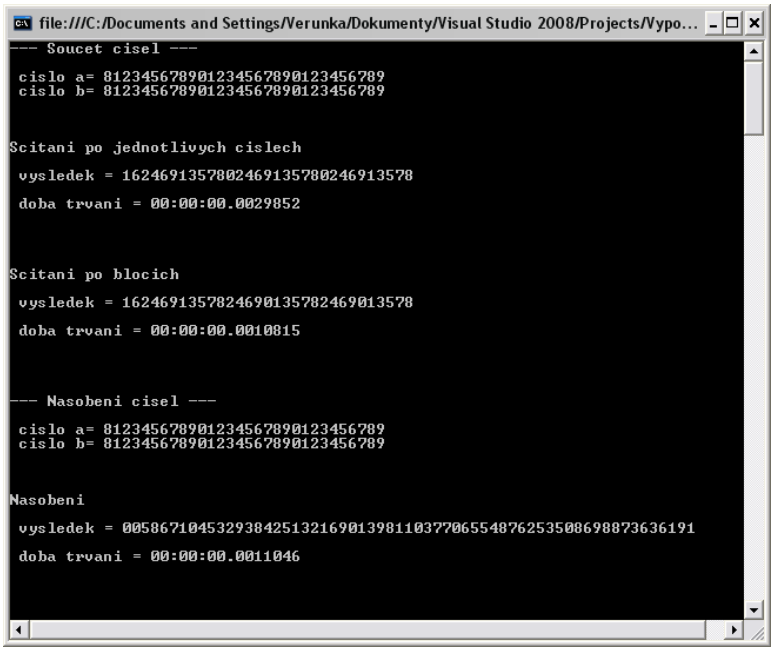
  preperso:

  siteID: 0444; date: 8340

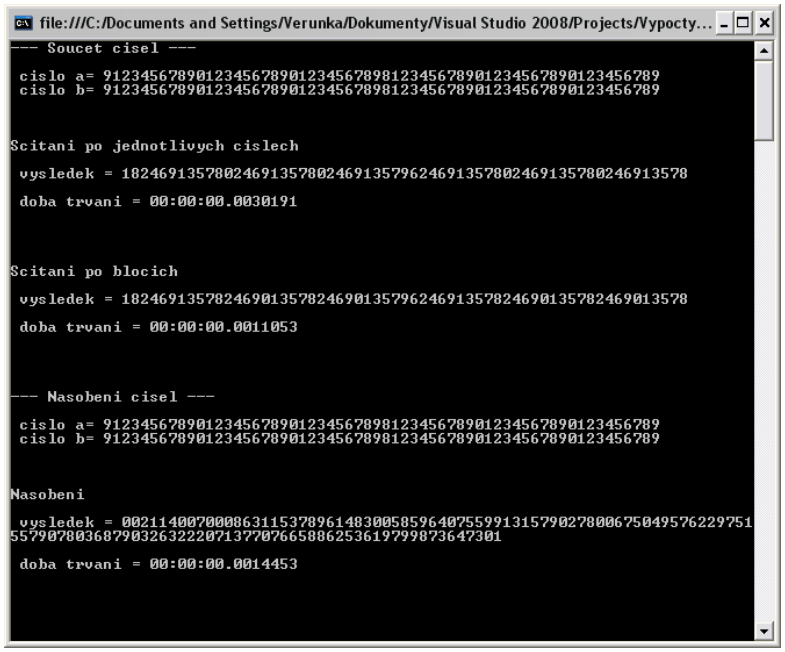
  equipmentID: 00010061

B DRUHÁ PŘÍLOHA

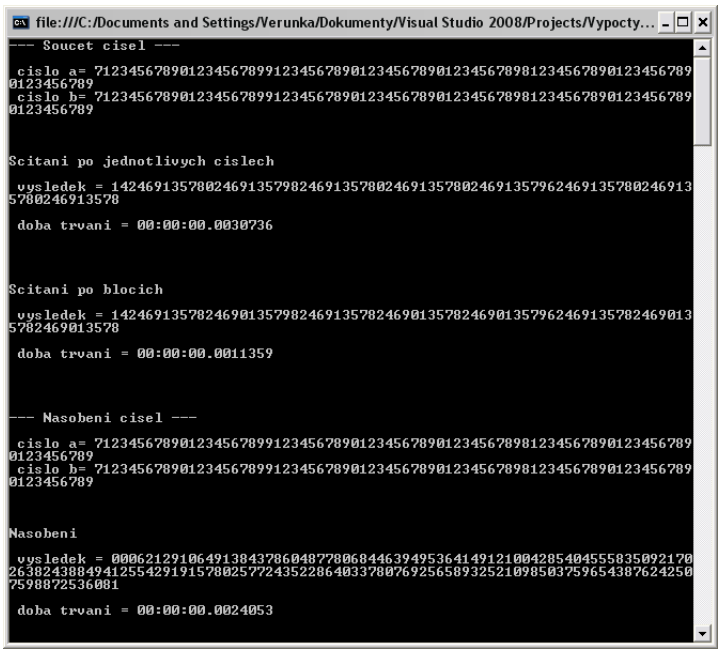
B.1 Výsledky simulace matematických operací



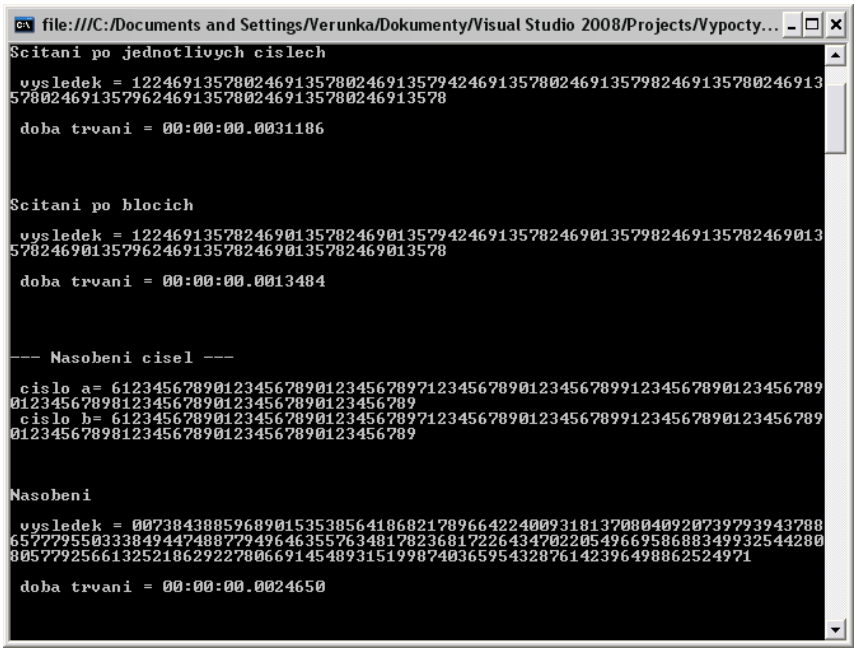
Obrázek B.1: Simulace výpočtů pro čísla se stejným řádem o délce 30.



Obrázek B.2: Simulace výpočtů pro čísla se stejným řádem o délce 60.

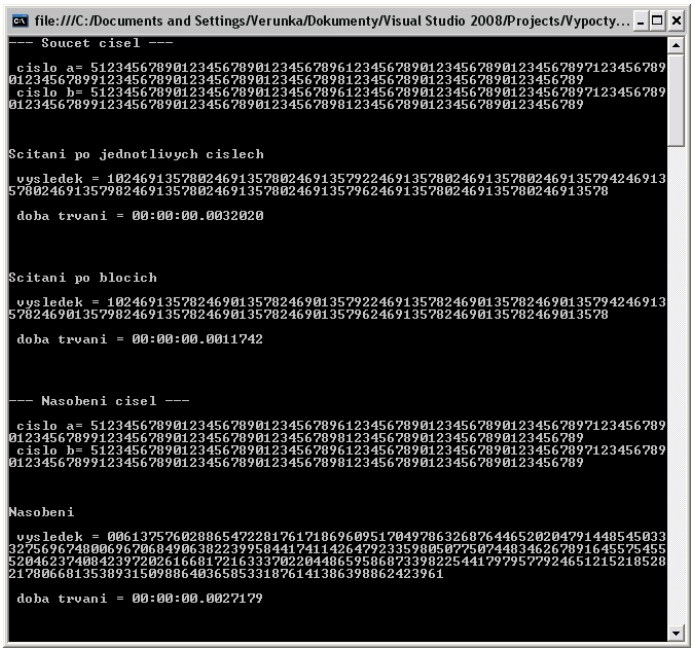


Obrázek B.3: Simulace výpočtů pro čísla se stejným řádem o délce 90.

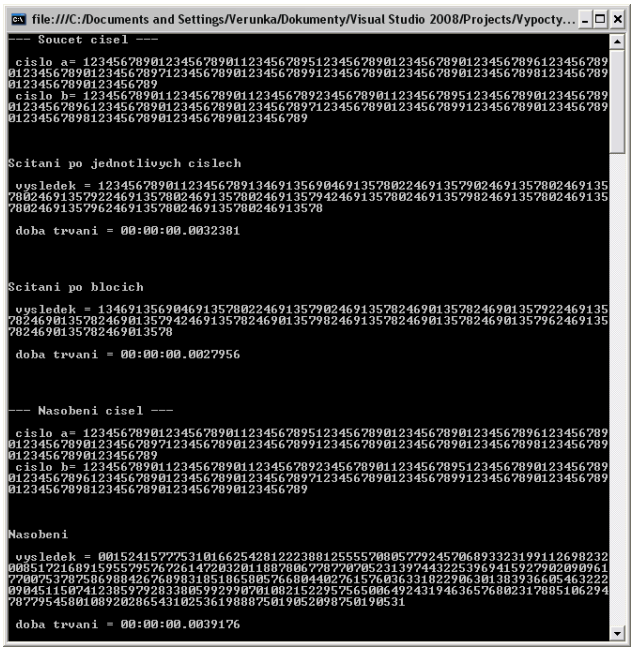


Obrázek B.4: Simulace výpočtů pro čísla se stejným řádem o délce 120.

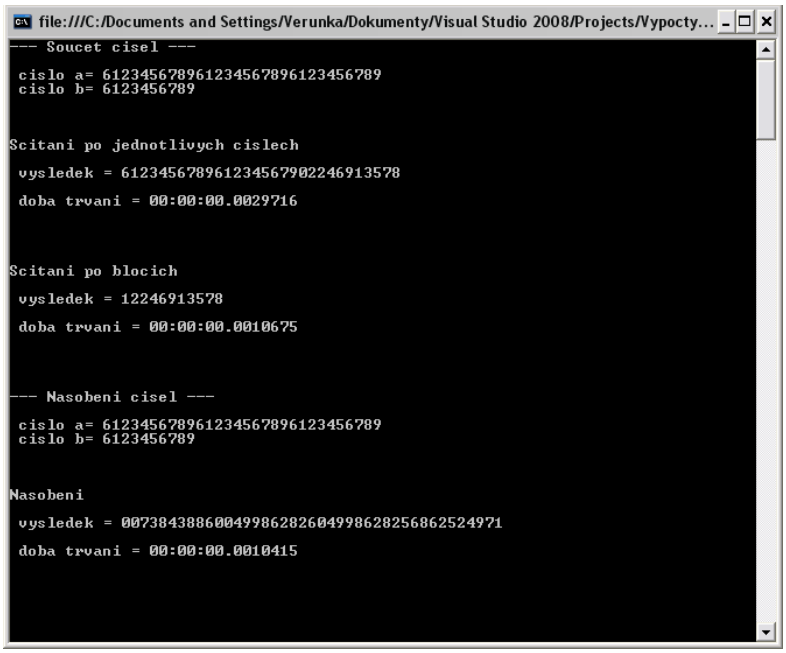
B.2 Graf závislosti času na počtů řádů čísel



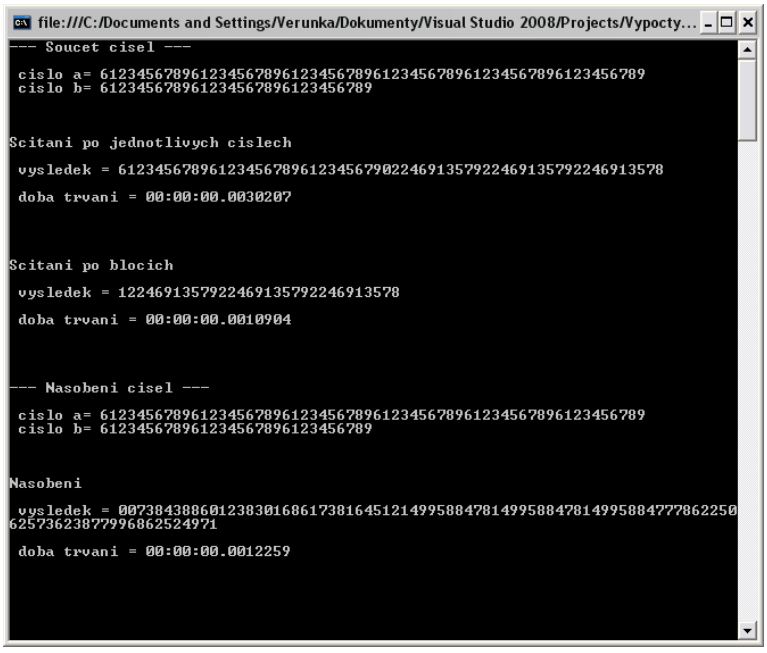
Obrázek B.5: Simulace výpočtů pro čísla se stejným řádem o délce 150.



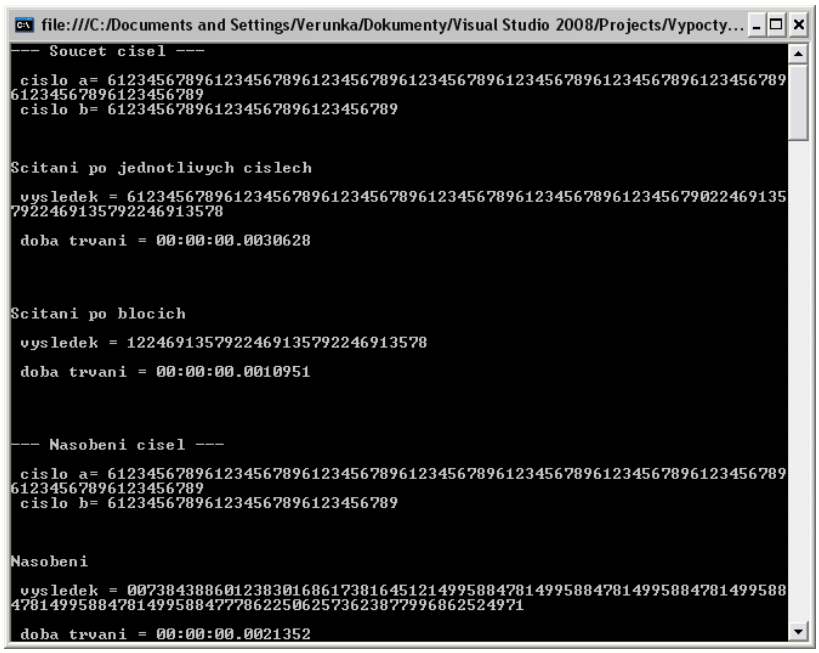
Obrázek B.6: Simulace výpočtů pro čísla se stejným řádem o délce 180.



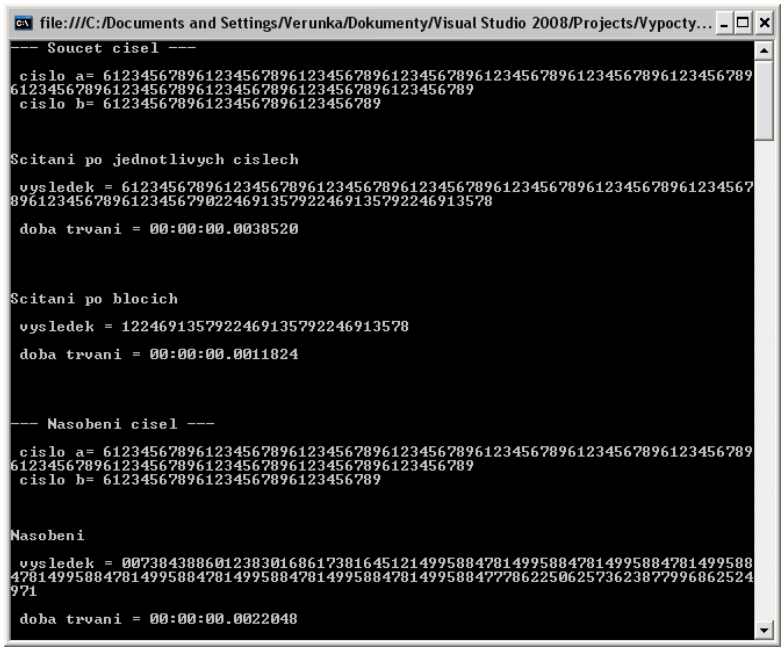
Obrázek B.7: Simulace výpočtů pro čísla s různými řády.



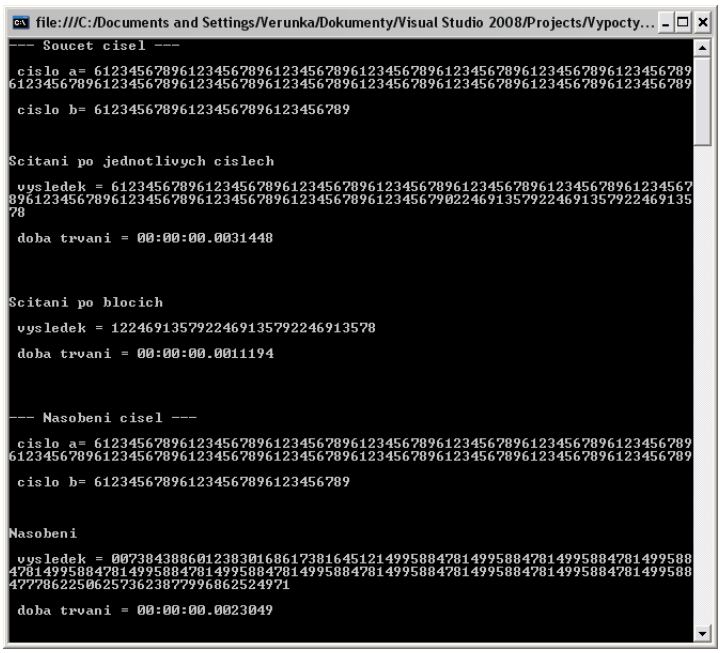
Obrázek B.8: Simulace výpočtů pro čísla s různými řády.



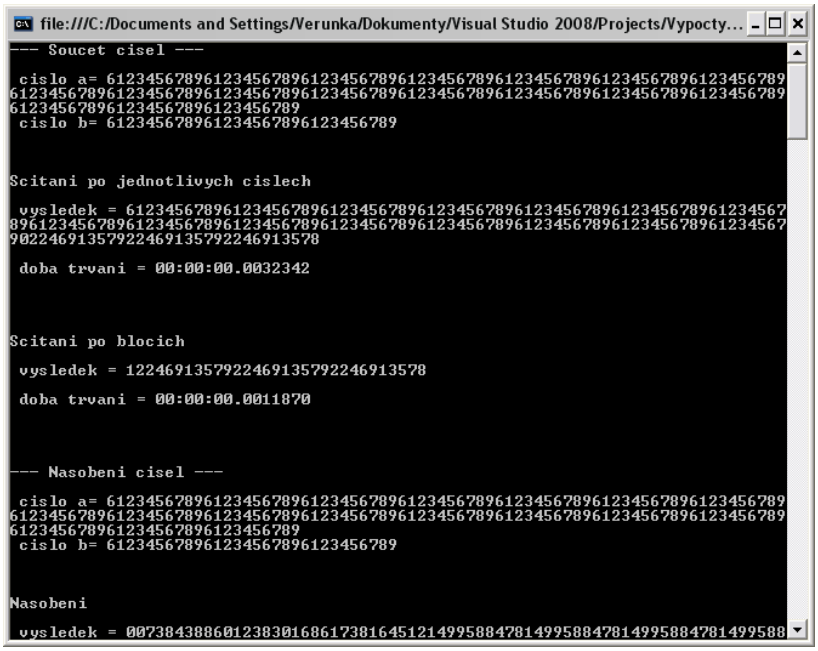
Obrázek B.9: Simulace výpočtů pro čísla s různými řády.



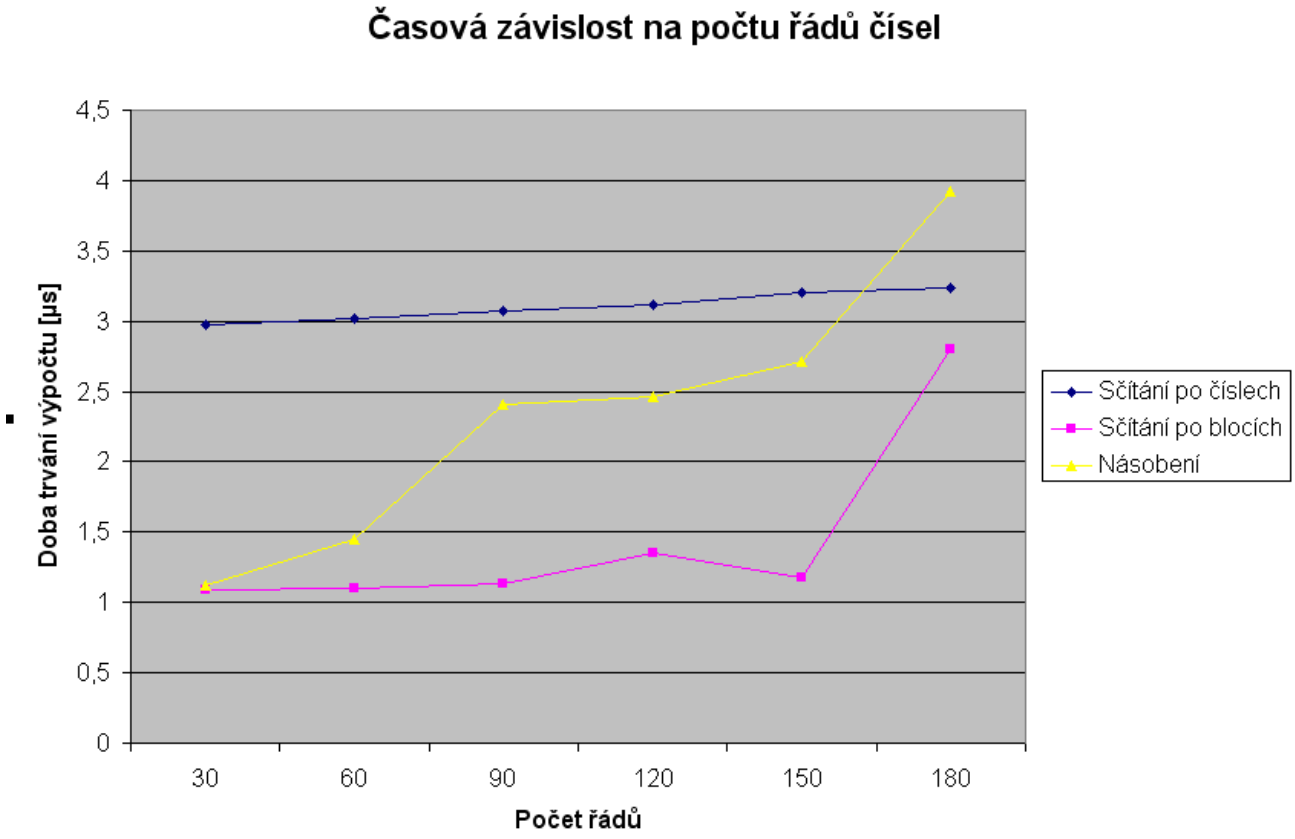
Obrázek B.10: Simulace výpočtů pro čísla s různými řády.



Obrázek B.11: Simulace výpočtů pro čísla s různými řády.



Obrázek B.12: Simulace výpočtů pro čísla s různými řády.



Obrázek B.13: Graf závislosti času na počtů řádů čísel.

C TŘETÍ PŘÍLOHA

Součástí diplomové práce je i CD, které obsahuje vytvořené kódy a aplikace. Na CD lze najít dvě složky - s názvy VypoctyConsole, a DP. Dále je součástí elektronická verze této práce.

Ve složce VypoctyConsole je k dispozici simulační aplikace, která umožňuje optimalizovaně sčítat a násobit různě dlouhá čísla. Pro otevření této aplikace je nutné mít nainstalován Microsoft Visual Studio, nejlépe verzi 2008.

Ve složce DP se nachází aplikace sloužící pro navázání komunikace klient-server (počítač a smartkarta). Pro spuštění těchto aplikací se doporučuje mít připojenou čtečku karet se samotnou smartkartou.