

Nástroje pro testování integrace PKI čipových karet do operačních systémů

Tools for PKI smart cards integration testing in operation systems

Bc. Miroslav Ingr

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Miroslav INGR**
Osobní číslo: **A09750**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Téma práce: **Nástroje pro testování integrace PKI čipových karet do operačních systémů**

Zásady pro vypracování:

1. Vypracujte rešerši dostupných zdrojů zabývajících se integrací PKI čipových karet do různých operačních systémů, případně aplikací.
2. Analyzujte odlišné přístupy integrace PKI čipových karet do operačních systémů, případně vybraných aplikací.
3. Pro alespoň dva různé operační systémy, popř. aplikace, které využívají odlišný způsob integrace PKI čipové karty, navrhnete sadu testů integračního rozhraní.
4. Navrhnete a vytvořte nástroj, pomocí něž je možné realizovat testy integračních rozhraní.
5. S pomocí testovacího nástroje provedte testy minimálně dvou různých integračních rozhraní PKI čipových karet.
6. Provedte rozbor získaných výsledků.
7. V závěru popište možnost použití navrženého nástroje pro testování společného multiplatformního rozhraní, které by bylo schopno integrovat PKI čipové karty do různých operačních systémů.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. RANKL, Wolfgang – EFFING, Wolfgang. Smart Card Handbook, 4. vydání, John Wiley & Sons, Ltd, 2010, 0-470-74367-0
2. ČSN ISO/IEC 7816-4 (369205), Identifikační karty – Karty s integrovanými obvody – Část 4: Organizace, bezpečnost a příkazy pro výměnu, leden 2006
3. ISO/IEC 7816-8:2004, Identification cards – Integrated circuit cards – Part 8: Commands for security operations, Edition: 2
4. ČSN ISO/IEC 7816-15 (369205), Identifikační karty – Karty s integrovanými obvody – Část 15: Aplikace kryptografické informace, listopad 2004
5. ČSN EN 14890-1 (369710), Aplikační rozhraní pro čipové karty používané jako zařízení pro vytváření bezpečného podpisu – Část 1: Základní služby, březen 2010
6. ČSN EN 14890-2 (369710), Aplikační rozhraní pro čipové karty používané jako zařízení pro vytváření bezpečného podpisu – Část 2: Další služby, 2010
7. ISO/IEC 24727-1:2007, Identification cards – Integrated circuit card programming interfaces – Part 1: Architecture, Edition: 1
8. ISO/IEC 24727-2:2008, Identification cards – Integrated circuit card programming interfaces – Part 2: Generic card interface, Edition: 1

Vedoucí diplomové práce: **prof. Ing. Karel Vlček, CSc.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **24. února 2011**

Termín odevzdání diplomové práce: **18. května 2011**

Ve Zlíně dne 24. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Diplomová práce se zabývá technologickými předpoklady pro integraci PKI čipových karet na platformách Microsoft Windows, Linux a Mac OS X. Použitím navrženého testovacího nástroje je realizována sada testů. Verifikovány jsou vlastnosti systémových rozhraní PKCS #11 a Microsoft CNG: KSP. Závěr práce hodnotí dosažené výsledky a jejich uplatnění v praxi.

Klíčová slova:

CDSA, CNG, CSP, JavaCard, KSP, čipová karta, minidriver, Mozilla, PKCS #11, PKCS #15, PKI, OpenSC, TokenD, zaručený elektronický podpis

ABSTRACT

Diploma thesis describes technological prerequisites for PKI smart cards integration on platforms Microsoft Windows, Linux and Mac OS X. A testing set is realized by using designed testing tool. Interface properties are verified for technologies PKCS #11 and Microsoft CNG: KSP. The thesis conclusion rates achieved results with practice usage.

Keywords:

CDSA, CNG, CSP, digital signature, JavaCard, KSP, minidriver, Mozilla, PKCS #11, PKCS #15, PKI, OpenSC, smart card, TokenD

Rád bych poděkoval panu prof. Ing. Karlovi Vlčkovi, CSc. za cenné rady, připomínky a vedení při tvorbě diplomové práce. Děkuji také Ing. Jiřímu Gieslovi, Ph.D. a Mgr. et Bc. Lukáši Josefíkovi za technické připomínky a korekturu textu.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

Úvod	9
TEORETICKÁ ČÁST	10
1 IDENTIFIKACE A AUTENTIZACE	11
2 KRYPTOGRAFIE	12
2.1 <i>Historické metody</i>	12
2.1.1 Substituční šifry	12
2.1.2 Aditivní šifry	13
2.1.3 Tranzpoziční šifry	13
2.1.4 Kombinované šifry	13
2.2 <i>Symetrická kryptografie</i>	14
2.3 <i>Asymetrická kryptografie</i>	15
3 PKI – PUBLIC KEY INFRASTRUCTURE	17
3.1 <i>Certifikační autorita</i>	17
3.2 <i>Modely důvěry</i>	18
3.2.1 Hierarchický model	18
3.2.2 Distribuovaný model (Web of Trust)	19
3.2.3 Přímý model (Peer to Peer)	19
4 ELEKTRONICKÝ PODPIS A CERTIFIKÁTY	20
4.1 <i>Elektronický podpis</i>	20
4.1.1 Princip funkce	21
4.1.2 Postup při vytváření a ověřování digitálního podpisu	22
4.2 <i>Digitální certifikát</i>	23
4.2.1 Formát a struktura certifikátu	24
4.2.2 Speciální případy certifikátu	24
4.2.3 Užití certifikátu	24
4.2.4 Známé certifikační autority	24
5 ČIPOVÁ KARTA	25
5.1 <i>Tělo karty</i>	26
5.2 <i>Technologický přehled</i>	26
5.3 <i>Využití čipových karet</i>	28
5.4 <i>Čtečka čipových karet</i>	28
5.5 <i>Komunikační protokol</i>	29
6 PROGRAMOVÁ ROZHŘANÍ PKI	31
6.1 <i>PKCS</i>	31
6.1.1 PKCS #11	32
6.1.2 PKCS #15	34
6.2 <i>Cryptography API</i>	35
6.2.1 CSP	36
6.3 <i>Cryptography API: Next generation</i>	38
6.3.1 CNG – Key Storage Provider	39
6.3.2 CNG – Minidriver	40
6.4 <i>CDSA</i>	43
6.4.1 CSSM API	44
6.4.2 TokenD	45
6.5 <i>Java Card</i>	45
6.6 <i>OpenSC</i>	47
6.7 <i>Mozilla</i>	48
6.8 <i>Další nativní implementace</i>	49

PRAKTICKÁ ČÁST	50
7 TESTY ROZHRANÍ	51
7.1 Test PKCS #11	51
7.2 Test CryptoAPI – CSP	53
7.2.1 Test základních kryptografických funkcí	53
7.3 Test Smart Card Base CSP/KSP - Minidriver	54
7.4 Test CNG – KSP	55
8 ROZBOR VÝSLEDKŮ	56
8.1 Testy PKCS #11	56
8.1.1 Test 1 – Podporované funkce	56
8.1.2 Test 2 – Šifrovací algoritmy	56
8.1.3 Test 3 – Hashovací algoritmy	57
8.1.4 Test 4 – Algoritmy pro digitální podpis/MAC	57
8.2 Test CNG – KSP	57
8.2.1 Test 1 – Podporované funkce	58
8.2.2 Test 2 – Podporované algoritmy	58
ZÁVĚR	59
CONCLUSION	61
SEZNAM POUŽITÉ LITERATURY	63
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	66
SEZNAM OBRÁZKŮ	67
SEZNAM TABULEK	68
SEZNAM PŘÍLOH	69

ÚVOD

Infrastruktura PKI výrazně přispívá ke zvyšování bezpečnosti digitálních informací. Její praktické využití nalezneme v bankovníctví, komunikačních technologiích, řízení přístupu a dalších odvětvích. PKI ve společné množině zahrnuje hardware, software, lidské zdroje, politiky a služby pro vytváření, správu, distribuci, skladování a zneplatnění digitálních certifikátů. Bezpečnost je založena na modelu asymetrické kryptografie. Důležitou roli zde hraje certifikační autorita. Veřejný klíč je svázán s identitou uživatele, která musí být v jedné certifikační doméně unikátní. Jeho hodnota může být veřejně čitelná. Soukromý klíč naopak nesmí být zveřejněn, a proto je nutné jej bezpečně uchovávat a chránit. K tomu se s úspěchem již řadu let používají čipové karty a další bezpečný hardware.

Technologii čipových karet řadíme do spodní úrovně z pohledu celého konceptu PKI. Uživatel je identifikován při vložení karty do čtecího zařízení. Autenticita uživatele je ověřena prostřednictvím číselného kódu PIN, který je chráněn v paměti karty. Pro případného útočníka není snadné hodnotu vyčíst, protože karetní rozhraní přímo neumožňuje přístup k bezpečné schránce. Po zadání správného kódu může být uživatel přihlášen nebo může být provedena jiná podporovaná operace. Tou může být například klientská autentizace internetového protokolu SSL, zašifrování dat či vytvoření digitálního podpisu.

Právě ochrana citlivých dat je jednou z nejdůležitějších součástí systémů PKI. Současná kryptografie nabízí řadu ověřených šifrovacích algoritmů. V PKI aplikacích se často používají asymetrické šifry RSA, Diffie-Hellman a stále se rozšiřující uplatnění nalézají algoritmy pracující s eliptickými křivkami. Pro specifické účely šifrování dat se v PKI využívá také symetrická kryptografie. Ověření integrity a původu dat je možné použitím digitálního podpisu. Při ověřování se používá asymetrická kryptografie. V první fázi vytvoříme miniaturu (tzv. hash) cílových dat. V praxi se používají standardní hashovací algoritmy - MD5, SHA-1, SHA-2 a další. Vytvořený hash zašifrujeme soukromým klíčem. Hodnotu pak přiložíme společně s daty a digitálním certifikátem. V certifikátu jsou uloženy identifikační údaje subjektu, vydavatele a veřejný klíč k ověření integrity dat.

Na základě výše zmíněných principů lze zakládat rozsáhlé infrastruktury se zabezpečením na několika úrovních. Dostupná systémová rozhraní mohou výrazně usnadnit vývoj. U standardních rozhraní se vývojář může přidržet ustálených postupů a doporučení.[1][2]

I. TEORETICKÁ ČÁST

1 IDENTIFIKACE A AUTENTIZACE

Každý subjekt v reálném světě představuje originální entitu. Tu popíšeme dostatečným množstvím vlastností, například jménem, příjmením, datem narození, biometrickými údaji, atd. Subjekt tak získává vlastní identitu, kterou se může prezentovat před svým okolím a může prokazovat způsobilost k provádění určitých pravidly ohraničených činností ve společnosti. Informace jsou uchovány v tisknuté podobě na papíře, ale čím dál více se užívá také elektronického zpracování. Hovoříme o identifikačních průkazech. Mezi tři nejznámější patří občanský průkaz, řidičský průkaz a pas.

Proces identifikace hledá na základě vložené identity správný subjekt, ke kterému náleží. Přímo romantickým příkladem je muž hledající jméno půvabné ženy z fotografie na zdi kavárny někde v hloubi pařížských uliček. Od číšníka se pak muž dozvídá křestní přízvisko a adresu ateliéru, ve kterém byl snímek pořízen. Nakonec osud zavelí, protože dáma vstupuje do kavárny a muž již ví, jak ji správně oslovit. Fotografie může také přenášet a měnit identitu. Cílená manipulace s identitou může skrývat řadu negativních vlastností osobnosti, a proto se vždy snažíme pravost ověřit.

Autentizace se zabývá ověřením pravosti identity subjektu. U lidí se používají tyto metody:

1. **Vlastnictví** – doklad v peněžence, čipová karta, pečeť
2. **Znalost tajemství** – heslo, číselný kód, šifra
3. **Biometrické vlastnosti** – obličej, otisky prstů, oční duhovka

Každá z těchto metod má mnoho praktických řešení a dalších přístupů, které často využívají moderní matematiky a technologií. Jejich vzájemná kombinace pak výrazně zvyšuje bezpečnost a snižuje pravděpodobnost omylu.

2 KRYPTOGRAFIE

Kryptografie neboli šifrování je nauka o metodách utajování smyslu zpráv převodem do podoby, která je čitelná jen se speciální znalostí. Slovo kryptografie pochází z řečtiny – *kryptós* je skrytý a *gráphein* znamená psát. Někdy je pojem obecněji používán pro vědu o čemkoli spojeném se šiframi jako alternativa k pojmu kryptologie. Kryptologie zahrnuje kryptografii a kryptoanalýzu, neboli luštění zašifrovaných zpráv.

Kryptografie se po staletí vyvíjela k větší složitosti zároveň s lidskou civilizací a mnohokrát ovlivnila běh dějin. Zejména utajení či vyzrazení strategických vojenských informací může mít zásadní vliv. Ale také prozrazení politických intrik, přípravy atentátů nebo i jen prozrazení milenců a podobně, to vše může úzce záviset na bezpečném přenosu informací a na schopnostech protivníka šifru rozbít.[3]

2.1 Historické metody

Starší sestrou kryptografie je steganografie neboli ukrývání zprávy jako takové. Sem patří různé neviditelné inkousty, vyrývání zprávy do dřevěné tabulky, která se zalije voskem apod. V moderní době lze tajné texty ukrývat například do souborů s hudbou či obrázky namísto náhodného šumu.

2.1.1 Substituční šifry

Substituční šifra obecně spočívá v nahrazení každého znaku zprávy jiným znakem podle nějakého pravidla. Pravděpodobně nejstarší popis substituční šifry je v Kámasútře, která se datuje do 4. století, ovšem její autor čerpal z pramenů až o 800 let starších.[3]

Posun písmen

Caesarova šifra je pojmenovaná po Juliu Caesarovi, který ji pravděpodobně používal jako první. Každé písmeno tajné zprávy je posunuto v abecedě o pevný počet pozic. Šifra je z dnešního pohledu velmi snadno luštitelná, protože je jen málo možných klíčů. Ve své době ale představovala nevídanou metodu a osvědčila se velmi dobře.

Substituční tabulky

Šifrování pomocí substituční tabulky, které je založeno na záměně znaku za jiný bez jakékoli vnitřní souvislosti či na základě znalosti klíče (hesla).

2.1.2 Aditivní šifry

Vigenèrova šifra

Jedná se o speciální případ polyalfabetické šifry. Vigenèrova šifra používá heslo, jehož znaky určují posunutí otevřeného textu a to tak, že otevřený text se rozdělí na bloky znaků dlouhé stejně jako heslo a každý znak se sečte s odpovídajícím znakem hesla. Caesarova šifra je tedy speciálním případem Vigenèrovy šifry s heslem o délce jeden znak. Vigenèrova šifra způsobuje změny pravděpodobnosti rozložení znaků a tím v podstatě znemožňuje kryptoanalýzu na základě analýzy četnosti znaků v textu. Luštění je založeno na vyhledávání vzdálenosti bigramových či trigramových dvojic v šifrovém textu a určováním jejich společného dělitele vedoucí k zjištění délky hesla.[3]

Vernamova šifra

Vernamova šifra je anglicky často nazývaná *one-time pad*, v českém překladu *jednorázová tabulková šifra*. Jde dosud o jedinou známou šifru, o níž bylo exaktně dokázáno, že je nerozluštitelná. Podobně jako Vigenèrova šifra i tahle spočívá ve sčítání písmen otevřeného textu a hesla, avšak heslo je blok náhodně zvolených dat o stejné velikosti jako je otevřený text. Pokud jsou splněny podmínky zcela náhodného klíče o stejné délce jako zpráva sama (a není použit opakovaně), je tato šifrovací metoda absolutně bezpečná. Délka klíče však pro běžné účely použití této metody zpravidla znemožňuje, metoda se používá hlavně pro velice specializované účely, např. tzv. *horká linka* spojující za dob studené války Moskvu a Washington používala Vernamovu šifru. Dnes se využívá v kombinaci s kvantovou kryptografií.[3]

2.1.3 Tranzpoziční šifry

Transpozice neboli přesmyčka spočívá ve změně pořadí znaků podle určitého pravidla. Například tak, že otevřený text je zapsán do tabulky po řádcích a šifrový text vznikne čtením sloupců téže tabulky.[3]

2.1.4 Kombinované šifry

Nevýhody či slabiny jednotlivých šifer je možné alespoň částečně odstranit kombinací šifer. Je však potřeba vzít v úvahu, že kombinací stejných druhů šifer nedocílíme vyššího zabezpečení proti neoprávněné kryptoanalýze. Např. následné použití dvou tabulek záměn

k zašifrování textu je pro potencionálního útočníka to samé, jako použití jedné tabulky s jiným rozložením.[3]

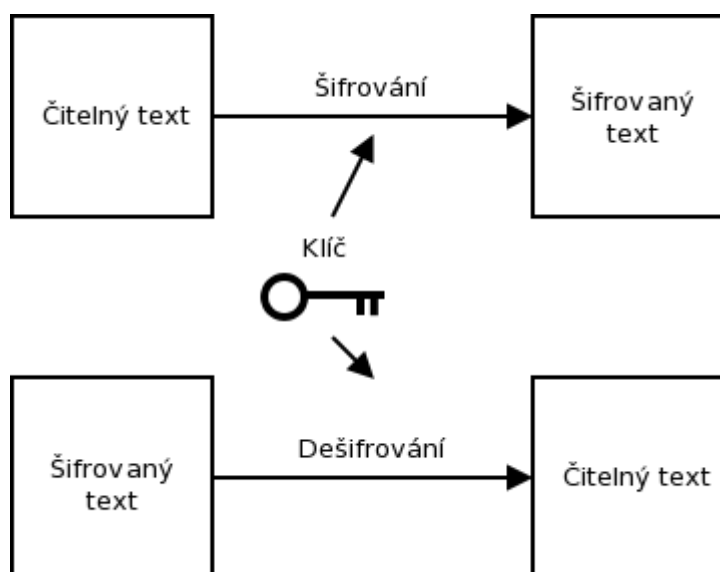
2.2 Symetrická kryptografie

Symetrické šifry používají tajný klíč, kterým je informace zašifrována. Odesílatel a příjemce zprávy se musí předem domluvit na tajném klíči, který musí oba dobře chránit proti prozrazení. Po zašifrování je bez znalosti klíče zpráva zcela nečitelná, ale její bezpečnost závisí na síle použitého algoritmu. Šifrování a dešifrování jsou vzájemně inverzní matematické operace. Díky znalosti klíče a použitého algoritmu si příjemce tajné zprávy již poměrně snadno zrekonstruuje původní text.

Hlavní výhodou symetrické kryptografie je bezesporu nízká výpočetní náročnost. V praxi se tato vlastnost využívá při zabezpečení objemných dat a tajný klíč je dodatečně zabezpečen pomocí bezpečnější asymetrické kryptografie.[4]

Symetrické šifry rozdělujeme na:

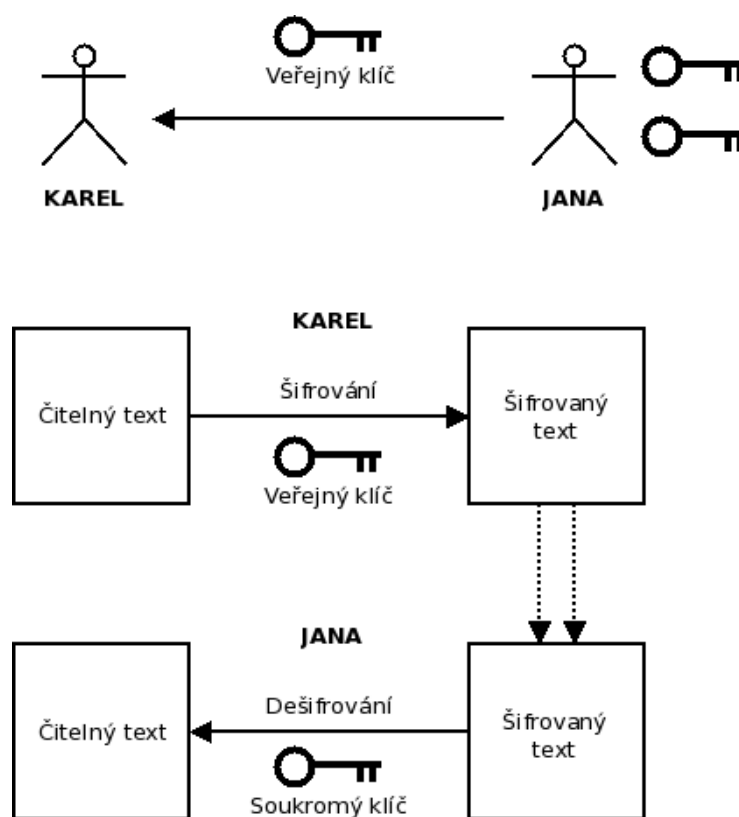
- **Proudové** – zpracování dat po jednotlivých bitech (FISH, RC4).
- **Blokové** – data jsou zpracována ve stejně velikých blocích (AES, Blowfish, DES, IDEA, RC2, RC5, TripleDES, Twofish a SkipJack).



Obr. 1: Princip symetrického šifrování

2.3 Asymetrická kryptografie

Asymetrické šifry používají na rozdíl od předešlého typu dva rozdílné šifrovací klíče. Nazýváme je soukromý a veřejný klíč. Příjemce si vygeneruje tento pár klíčů. Veřejný klíč zašle odesílateli zprávy, který jej použije k šifrování. Po přijetí nečitelné zprávy získá příjemce obsah díky svému soukromému klíči. Příjemce nemusí s odesílatelům sdílet žádné tajemství a stále spolu mohou bezpečně komunikovat pomocí šifrovaných zpráv.



Obr. 2: Princip asymetrického šifrování

Základem asymetrických algoritmů je vhodná matematická funkce, která má funkci inverzní s vysokou třídou složitosti. Jednoduchým příkladem je prosté násobení dvou prvočísel. Faktorizace výsledku na původní násobky představuje vysoké výpočetní nároky, protože zatím neexistuje algoritmus, který by dokázal v polynomiálním čase rozložit velké číslo na součin prvočísel. Z toho plyne, že lze získat klíče k šifře hrubou silou použitím inverzního algoritmu, ale bude to natolik časově náročné, že se nevyplatí čekat (například 10^{54} let). Vhodné a bezpečné klíče jsou většinou dvě prvočísla dostatečného řádu, právě aby vyhověla těmto požadavkům. Kromě operace násobení prvočísel implementují algoritmy také diskretní logaritmus a další matematické funkce.

První asymetrickou šifrou je algoritmus Diffie-Hellman (DH) vynalezený v roce 1976. DH umožňuje bezpečnou výměnu klíčů a užití rychlejší symetrické šifry na zabezpečeném spojení.[6] Další známá asymetrická šifra RSA (Rivest-Shamir-Adleman) byla publikována v roce 1978 a je dodnes užívaným standardem.[7] Alternativou může být algoritmus ElGamal, který ale bohužel není příliš efektivní, protože zdvojnásobí objem zašifrovaných dat.[8]

Užití eliptických křivek v kryptografii navrhli v roce 1985 Neal Koblitz a Victor S. Miller. Algoritmy používající eliptické křivky získávají stále více popularity díky vyšší bezpečnosti při použití kratšího šifrovacího klíče. Mezi známé implementace patří ECDH (bezpečná výměna klíčů), ECDSA (digitální podpis) a další. V roce 2005 na konferenci RSA vznikla rodina algoritmů Suite B, které výhradně používají ECC k výměně klíčů a generování digitálního podpisu.[9]

3 PKI – PUBLIC KEY INFRASTRUCTURE

Public Key Infrastructure (PKI) je systém založený na asymetrické kryptografii (použití veřejného klíče), certifikačních autoritách a digitálních certifikátech. PKI ovšem nelze chápat jako úzce zaměřený systém, naopak se jedná o širokou infrastrukturu služeb a technologií. Do PKI zahrnujeme hardware, software, lidské zdroje, zásady a postupy nutné k vytváření, správu, distribuci, použití, skladování a zneplatnění digitálních certifikátů.

PKI systémy obecně poskytují tři základní služby:

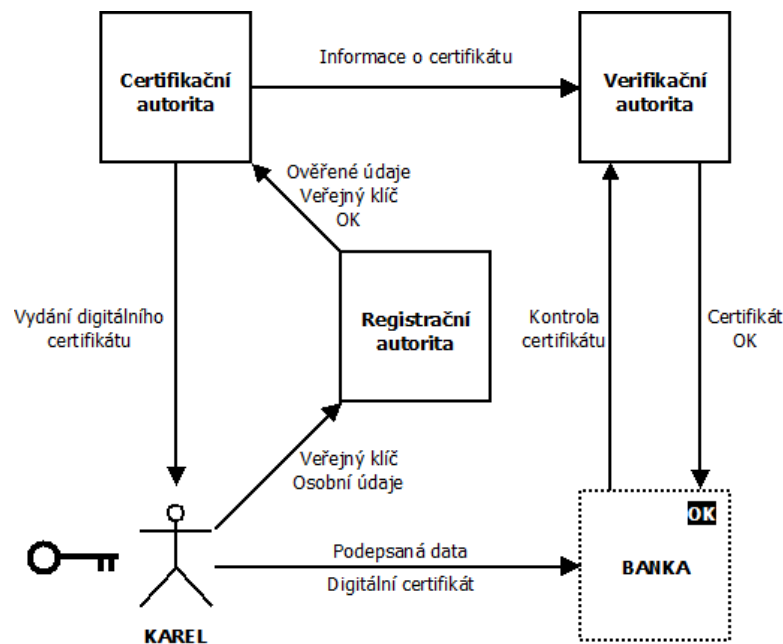
- **Autentizaci** – ověření identity uživatele,
- **Elektronický podpis** – integrita a původ dat,
- **Šifrování** – utajení dat.

Z pohledu kryptografie je PKI soustava, která váže veřejné klíče s odpovídajícími identitami uživatelů dle záměrů **certifikační autority** (CA). Identita uživatele musí být unikátní v rámci každé CA domény. Tato vazba je stanovená při registraci a vydání klíčů. Podle míry důvěryhodnosti je zajišťována buďto softwarem CA nebo lidským dohledem. Služba zajišťující vazby uživatelů s klíči se nazývá **registrační autorita** (RA). CA potom vydá digitální certifikát, ve kterém je zachována společně identita uživatele, veřejný klíč, jejich vazba, podmínky platnosti a další důležité vlastnosti pro dané použití.[1][2]

Na rozdíl od veřejného klíče patří soukromý klíč k vysoce citlivým položkám, proto je nutné tento klíč chránit proti potenciálnímu zneužití. Elegantní možností je užití „bezpečného“ hardware – USB krypto token, čipová karta, specializovaný server. Certifikát s veřejným klíčem je často na kartě či tokenu přítomen také, ale není nutné jej chránit.

3.1 Certifikační autorita

Certifikační autorita je důvěryhodný subjekt, který vydává na požádání digitální certifikáty. Organizační složkou CA může být registrační autorita, která nejdříve ověřuje osobní údaje žadatelů. CA na základě rozhodnutí RA vydá nebo nevydá certifikát. Zároveň garantuje pravost údajů v něm uvedených. Vše je podepsáno zaručeným elektronickým podpisem, aby bylo možné kdykoliv zkontrolovat integritu. CA se stará také o zneplatnění starých a „špatných“ certifikátů pomocí CRL seznamů.[2]



Obr. 3: Vydání a použití digitálního certifikátu

3.2 Modely důvěry

Implementace PKI vyžaduje analýzu obchodních cílů a důvěryhodných vztahů, které v daném prostředí existují. Správné uvědomění takových vztahů vede k ustálení celkového modelu důvěry, který PKI systém bude vyžadovat. Následující tři modely slouží ke srovnání různých přístupů.[2]

3.2.1 Hierarchický model

Hierarchický model představuje typickou implementaci PKI. Funguje na bázi stromové struktury certifikačních autorit. Struktura CA je předem dohodnuta a poskládána. Koncovým uživatelům pak CA vydává certifikáty na základě principu přenosu důvěry.

Při budování tohoto modelu vznikají kompromisy při určování umístění důvěryhodných bodů pro koncové entity PKI systému. Ve vrstvené hierarchii s mnoha CA je možné rozdělit míru rizika na menší díly, ale zase není snadné administrovat celek. Snadnou administraci dosáhneme použitím ploché hierarchie s jedinou CA, ale o to větší vzniká riziko prolomení její stability a následného narušení celkové zabezpečení.

3.2.2 Distribuovaný model (Web of Trust)

Distribuovaný model nemusí používat služby CA, a tedy nevyžaduje třetí důvěryhodnou stranu, která rozhoduje o identitách koncových uživatelů. Model používá certifikáty typu podepsán-sám-sebou a možnou atestaci těchto certifikátů plně důvěryhodnou CA. Uživatel vystaví svůj veřejný šifrovací klíč na veřejném, ale důvěryhodném místě. Odtud je možné si klíč stáhnout a je možné navázat bezpečnou komunikaci. Nejznámější implementací tohoto modelu je PGP (Pretty Good Privacy) a GnuPGP. PGP se stalo standardem internetového šifrování zpráv a v roce 1997 byla zavedena specifikace OpenPGP. Výhodou je možnost spolupráce s plně důvěryhodnou CA všemi skupinami v doméně (například interní CA v rámci organizace). Více informací o PGP v literatuře.[17]

3.2.3 Přímý model (Peer to Peer)

Přímý model důvěry se používá v systémech založených na symetrickém šifrování. Nepoužívá třetí důvěryhodnou stranu typu CA ani jiné. Každý koncový uživatel navazuje důvěryhodný vztah s ostatními uživateli individuálně. Model je vhodný spíše pro domácí nebo laboratorní účely, ale neposkytuje dostatek bezpečí do internetového komerčního světa.

4 ELEKTRONICKÝ PODPIS A CERTIFIKÁTY

4.1 Elektronický podpis

Pojem elektronický podpis má několik možných výkladů. V obecnějším pojetí se jedná o připojení podpisu, iniciále, kontaktních a jiných identifikačních údajů v elektronickém dokumentu. V užším pojetí hovoříme o **zaručeném elektronickém podpisu** (digitální podpis, z angl. digital signature), který je vymezen legislativou dané země. V České republice platí zákon o elektronickém podpisu (zákon č. 227/2000 Sb., o elektronickém podpisu), který předešlé pojmy vymezuje v rámci platné legislativy. Ve spojení s infrastrukturou PKI hovoříme o pojmu **digitální podpis** a posuzujeme jej z pohledu číslicových technologií.[1][10]

Digitální podpis je matematické schéma prokazující pravost a integritu digitálního dokumentu nebo zprávy. Matematický princip je založen na asymetrické kryptografii a použití veřejného/soukromého klíče. Díky tomu může odesílatel dát příjemci zprávy najevo, že přišla zaručeně nepoškozená a od autentického odesílatele. Tento mechanismus se běžně užívá při distribuci software, bankovních transakcích, a je také důležitý při odhalování padělků. Pro některé účely je navíc vyžadován tzv. **kvalifikovaný elektronický podpis** pouze s předepsanými typy certifikace, tedy je založený na kvalifikovaném certifikátu, o kterém pojednává část *Speciální případy certifikátu*. Zaručený elektronický podpis dokumentu zajišťuje:

- **autenticitu** – lze ověřit původnost, identitu subjektu,
- **integritu** – lze prokázat, že po podepsání nedošlo k žádné změně, soubor není úmyslně či neúmyslně poškozen,
- **nepopiratelnost** – autor nemůže tvrdit, že podepsaný elektronický dokument nevytvořil (např. nemůže se zříct vytvoření a odeslání výhružného dopisu),
- může obsahovat **časové razítko**, které prokazuje datum a čas podepsání dokumentu.

Rozdíl mezi prostým a zaručeným elektronickým podpisem je obdobný rozdílu mezi úředně neověřeným a ověřeným vlastnoručním podpisem, přičemž možnost, obtížnost a spolehlivost písmo-znaleckého rozboru neověřeného vlastnoručního podpisu lze přirovnat k možnosti, obtížnosti a spolehlivosti ověření autenticity nezaručeného elektronického podpisu.[11]

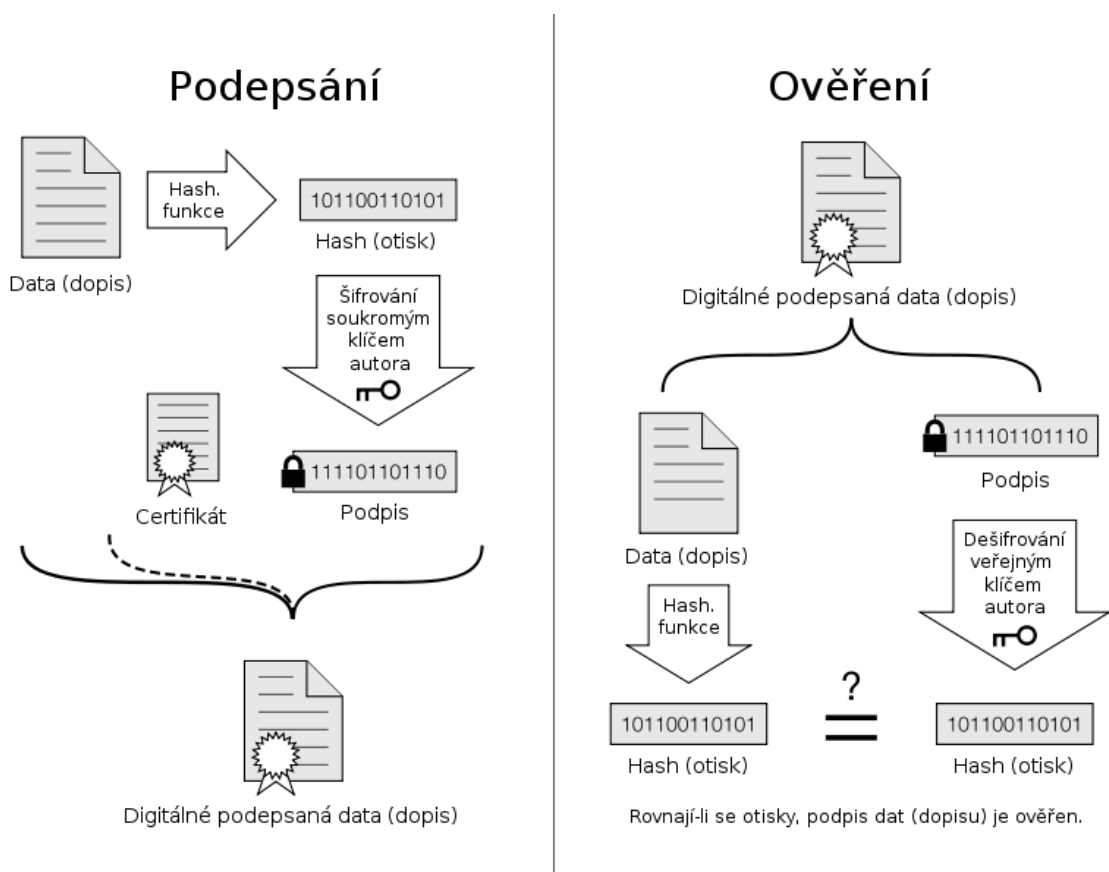
4.1.1 Princip funkce

Zaručený elektronický podpis je aplikací asymetrické kryptografie (tj. kryptografie s veřejným klíčem). Výjimečně může k jeho vytvoření posloužit i symetrická kryptografie, pak se ovšem jedná o arbitrovaný protokol. Principem běžného elektronického podpisu je tedy zašifrování (miniatury) dokumentu soukromým klíčem a jeho následné dešifrování veřejným klíčem. Soukromý a veřejný klíč si můžeme představit jako dvě velmi vysoká čísla. Aby bylo šifrování účinné, nesmí být znám postup, pomocí kterého by bylo možné, ani s nasazením nejdokonalejších počítačů, vypočítat ze známé části zbývající soukromou část. Kdo nezná soukromý klíč, není schopen zašifrovat dokument (nebo miniaturu) tak, aby jej bylo možné dešifrovat veřejným klíčem. Soukromý klíč tedy nesmí znát nikdo jiný, než autor dokumentu.

Z praktických důvodů se místo celého dokumentu šifruje pouze jeho hash - což je jakási digitální obdoba otisku prstů. Při vytváření hashe pro šifrovací účely se používají takové algoritmy, že je prakticky vyloučené změnit dokument takovým způsobem, aby se současně nezměnil jeho hash. Příkladem jednoduchého hashe (pro kryptografii nepoužitelného) je kontrolní součet řady čísel, používaný pro ověření, že při jejich přepisování nedošlo k chybě. Hash je většinou mnohem kratší než celý dokument. Výhodou šifrování hashe místo celého dokumentu je čitelnost dokumentu i bez klíče a vyšší rychlost šifrování a dešifrování, která dovoluje použití složitějších (a tedy bezpečnějších) šifrovacích algoritmů. Prakticky se používají šifrovací algoritmy RSA či DSA v kombinaci s hashovacími funkcemi MD5 a SHA.[11]

4.1.2 Postup při vytváření a ověřování digitálního podpisu

Nejprve se vytvoří hash dokumentu, což je poměrně krátký (typicky několik stovek bitů) výtah vytvořený specializovaným algoritmem z celého dokumentu. Tento hash se poté zašifruje autorovým tajným klíčem, čímž vznikne podpis. Ověření podpisu pak spočívá v dešifrování hashe (podpisu) pomocí veřejného klíče autora, nezávislého výpočtu hashe z dokumentu a porovnání obou hodnot. Pokud si odpovídají, pak je podpis ověřen a dokument je považován za důvěryhodný. Autor nemůže popřít své autorství - pokud k jeho tajnému klíči nikdo jiný nemá přístup, pak nikdo jiný nemůže zašifrovat hash dokumentu tak, aby po aplikaci autorova veřejného klíče vznikla správná hodnota. Pokud by byl dokument po podepsání změněn nebo poškozen, vyšla by jiná hodnota hashe, takže elektronický podpis by byl neplatný.[11]



Obr. 4: Princip digitálního podpisu [11]

4.2 Digitální certifikát

Digitální certifikát nese identitu jeho majitele a také obsahuje digitálně podepsaný veřejný klíč, kterým je možné navázat komunikaci s majitelem soukromého klíče. Certifikát je uchován ve formátu dle standardu X.509, což je ITU-T používaný v PKI (infrastruktura založená na veřejných klíčích) pro tzv. jednotné přihlášení (z angl. single sign-on) a správu uživatelských práv PMI.

Důvěryhodnost certifikátů zaručuje **certifikační autorita**, která vydává certifikáty za úplaty a na požádání. Před vydáním je však nutné ověřit totožnost „zákazníka“ a vymezit platnost certifikace. Tzv. **kořenový certifikát** je vydán větší organizací a následně jsou podepsány také certifikáty koncových uživatelů, což mohou být například zaměstnanci. Každý koncový uživatel pak vlastní certifikát, který může také vymezovat jeho práva v rámci společnosti nebo PKI systému. Takto vzniká hierarchická certifikační cesta, která umožňuje systém přenosu důvěry mezi certifikáty.

Tento systém se hojně používá při zabezpečeném síťovém přenosu SSL. Většina internetových prohlížečů používá vlastní úložiště kořenových certifikátů, díky kterým je možné ověřit certifikát vzdáleného serveru. Tak klient ví, že server není podvodný a je navázáno šifrované spojení klient-server (například pomocí protokolu HTTPS).

X.509 také podporuje systém zneplatnění certifikátů v případě prozrazení soukromého klíče, případně prošlé platnosti, apod. Na internetu jsou ke stažení tzv. CRL, jedná se o seznamy již neplatných certifikátů. Aplikacím je silně doporučeno si dle standardu X.509 zkontrolovat platnost, pokud je odkaz na CRL v těle certifikátu uveden.

Důležitou vlastností každého certifikátu je jeho platnost, která se omezuje oboustranně jako „neplatný před“ a „neplatný po“. Doba platnosti nesmí být delší než je zaručená doba bezpečnosti použitého šifrovacího algoritmu. Také doba bezpečné úschovy citlivých údajů (soukromé klíče) hraje roli a je dobré s ní počítat. V opačném případě nelze považovat certifikát za bezpečný a tedy většinou ani platný. Síla použitého algoritmu by měla být dostatečná dle současných regionálně uznávaných standardů.

4.2.1 Formát a struktura certifikátu

Jednotlivé položky jsou seřazeny dle specifikace X.509 v datových strukturách ASN.1[21] a celek je zaveden do binární podoby ve formátu DER a CER (obojí založené na základním formátu BER), navíc se často užívá kódování BASE64. Formát X.509 již prošel třemi verzemi, každá do výsledné struktury přidává nová informační pole. Třetí verze X.509 v3 umožňuje do certifikátu vkládat specifické informace pomocí tzv. rozšíření (z angl. extensions). Formátu těchto rozšíření často rozumí pouze některé aplikace, společným znakem je pouze uložení dat ve strukturách ASN.1. Více o přesném znění formátu X.509 certifikátů se lze dočíst v literatuře [12][13].

4.2.2 Speciální případy certifikátu

Zvláštním případem je certifikát typu podepsán-sám-sebou (z angl. self-signed). Takový stav nastává, když je vydavatel zároveň subjektem. Důvodem k vydání takového certifikátu mohou být hlavně testovací účely při vývoji aplikací na bázi PKI systémů.

Dalším typem je **kvalifikovaný certifikát**, který je v České republice vymezen *Zákonem o elektronickém podpisu* (zákon č. 227/2000 Sb.).[10] Zákon přesně stanovuje, které vlastnosti a informace certifikát obsahuje. Také vymezuje užití certifikátu při komunikaci se státními institucemi, kde je uznáván podobně jako třeba cestovní pas.

4.2.3 Užití certifikátu

Hlavní užitek přinášejí certifikáty díky možnosti zaručeného elektronického podpisu, který lze používat při elektronické korespondenci, ověření pravosti softwarových produktů, atd. Dalším využitím je při navazování zabezpečené komunikace klient-server (například technologie SSL). V neposlední řadě hrají certifikáty důležitou roli při vymezování uživatelských pravomocí v rámci korporátního PKI systému.

Certifikát obsahuje také užitečné informace o jeho majiteli: iniciály a kontaktní údaje. Často bývá vymezen účel použití a v jakých aplikacích se může certifikát vyskytovat.

4.2.4 Známé certifikační autority

Certifikační autorita je zpravidla komerční a vydává různé druhy certifikace pro různé druhy uživatelů. Mezi nejznámější patří VeriSign, Thawte, I. CA, PostSignum,...

5 ČIPOVÁ KARTA

Čipová, chytrá karta neboli ICC (zkr. Integrated Circuit Card) se vyznačuje kapesní velikostí a vloženým integrovaným elektrickým obvodem. Jednou ze dvou hlavních kategorií ICC karet jsou čipové karty s vlastním mikroprocesorem a operační pamětí. Druhou jsou méně „chytré“ paměťové karty s již předem určenou bezpečnostní logikou. Z jiného pohledu můžeme dělit technologii na **kontaktní** s elektrickými vývody a **bezkontaktní** s integrovanou anténou.

Typickým příkladem paměťového typu jsou telefonní karty určené k zaplacení telefonního hovoru s předem nastaveným limitem volných jednotek. Dále se jedná o karty slevové, případně členské s podporou nadstandardních služeb. Práce se již více tímto typem nezabývá a mnohem hlouběji rozebírá ICC karty s vloženým mikroprocesorem.

Přítomnost mikroprocesoru s programovatelnou pamětí umožňuje aplikaci moderní kryptografie. Komunikace mezi čipem a počítačem probíhá díky standardnímu rozhraní PC/SC. Karta je fyzicky spojena s počítačem pomocí karetní čtečky a elektrickým kontaktem s vývody čipu (kontaktní typ) nebo probíhá komunikace bezdrátově pomocí antény a speciální bezkontaktní čtečky.

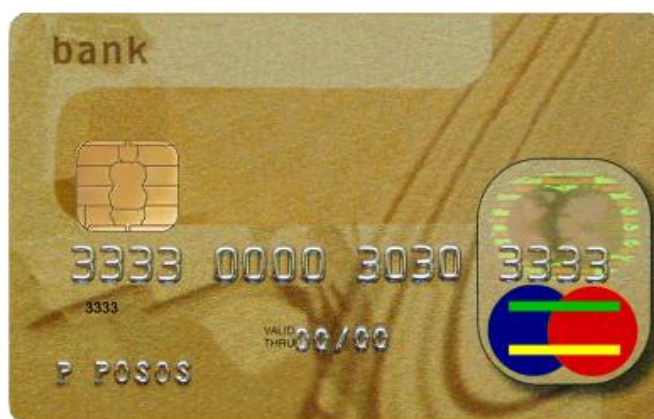
Čipové karty chápeme jako „bezpečný“ hardware, který je schopen chránit uložené citlivé údaje. Hlavním bezpečnostním prvkem je Personal Identification Number, zkráceně kód PIN. Jedná se tajnou kombinaci číslic (případně ASCII) o délce čtyř až osmi znaků, kterou by měl znát pouze oprávněný vlastník karty. Výhodou čtyřmístného čísla je dobrá zapamatovatelnost. Zadavatel je také často limitován omezeným počtem pokusů. V případě znalosti kódu je úspěšně autentizován. V opačném případě je PIN zablokován a nelze již další pokus provést. Odblokování lze provést pomocí „záložního“ kódu PUK, který se běžně skládá z osmi číslic. Po zadání správného kódu PUK je PIN odblokován, ale musí být znova nastavena jeho nová hodnota.

Externí provedení karty může být čisté (obvykle bílá barva), s jednoduchým popisem nebo s barevným potiskem. Jednoduchý popis tvoří černobílé znaky s nejdůležitějšími údaji. Karta s barevným potiskem může být vybavena logem společnosti, organizace, jménem vlastníka, atd. Pro lepší identifikaci může být potisknuta také fotografií a dalšími personálními údaji. Potisk zvyšuje také estetickou hodnotu karty a některé typy výrazně zvyšují prestiž majitele.

5.1 Tělo karty

Norma ISO/IEC 7810 definuje nominální rozměry karty (ID-1) 85,60 x 53,98 mm. Tento standard je užíván většinou u identifikačních a platebních karet. Existuje také druhý populární standard (ID-000) 25 x 15 mm, který díky kompaktním rozměrům vyhovuje užití v mobilních telefonech (tzv. SIM karta) a jiných elektronických zařízeních. Oba typy mají stejnou tloušťku 0.76 mm.[14]

Tělo karty se vyrábí obecně z plastu, běžně z polyvinyl-chloridu, acrylonitril-butadien-styrenu nebo polykarbonátu. Oproti papírovým předchůdkyním dává užití plastu moderním kartám vyšší odolnost a trvanlivost. Karty mohou být barevně offsetově potisknuty pro zvýšení uživatelského komfortu i estetické hodnoty.



Obr. 5: ID-1 čipová karta [14]

5.2 Technologický přehled

Rozeznáváme dva typy „chytrých“ čipových karet: kontaktní a bezkontaktní. Oba typy mají mikroprocesor a paměť. Procesor disponuje vlastním operačním systémem a umožňuje zvládat více aplikací najednou (platební karta, předplacená členská karta a například také přístupová karta). Kontaktní a bezkontaktní technologie se liší pouze ve způsobu, jakým mikroprocesor komunikuje s vnějším světem.

Kontaktní typ má 8 vývodů, které se musí fyzicky dotýkat vývodů karetní čtečky, aby mohl být navázán přenos informace. Protože musí být kontaktní karty zasunuty do čtecího zařízení a musí dodržet správný směr kontaktu, nejsou z důvodu rychlosti a komfortu přijatelné pro většinu přístupových aplikací.



Obr. 6: Vývody kontaktní čipové karty [14]

Vývod (PIN)	Popis
C1 - VCC	Zdroj napájení
C2 - RST	Signál RESET, užívaný při resetování komunikace
C3 - CLK	Signál CLOCK, časování komunikace
C5 - GND	Uzemnění GROUND
C6 - VPP	Programovací napájení, dnes již zastaralé
C7 - I/O	Sériový vstup a výstup (poloduplexní)
C4, C8 - AUX	AUX1, AUX2 jsou užívaný pro rozhraní USB a další účely

Tab. 1: Vývody kontaktní čipové karty

Bezkontaktní typ používá rádiového přenosu, stejně jako například RFID identifikační karty. Používají ovšem rozdílnou frekvenci 13.56 MHz (namísto 125 kHz). Vyšší frekvence umožňuje přenášet rychleji a paralelní komunikaci s více kartami. Většina přístupových systémů čte pouze sériová čísla z karty a nepoužívá interní paměti. Ta však může být využita k uložení biometrických údajů vlastníka (markanty otisků prstů či oční sítnice). V takovém případě biometrická čtečka první přečte předlohu na kartě a poté ji porovnává s aktuálními údaji subjektu. Touto cestou nemusí být biometrická data distribuována a skladována v paměti HW (server, čtečka), což výrazně zjednodušuje globální systém a snižuje paměťové nároky.

5.3 Využití čipových karet

Čipové karty slouží k následujícím účelům:

- **Identifikace** – potisk karty osobními údaji, může obsahovat digitální certifikát.
- **Autentizace** – ověření pravosti uživatele pomocí kódů PIN/PUK.
- **Datové úložiště** – čip je vybaven interní zabezpečenou pamětí.
- **Aplikační zpracování** – spolupráce v rámci rozsáhlejších aplikací (PKI).

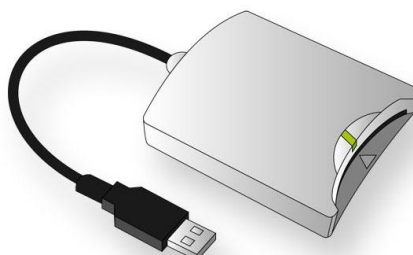
Nasazení karet pokrývá mnoho různých sektorů lidské činnosti:

- **Bankovníctví** - platební karty,
- **Telekomunikace** - SIM karta, předplacená telefonní karta,
- **Proprietární aplikace** – zákaznické/zaměstnanecké karty,
- Zabezpečení přístupu v budovách.

5.4 Čtečka čipových karet

Čtečka čipových karet je elektrické zařízení, které umožňuje komunikaci s vloženou/cílovou bezkontaktní kartou. Čtečky typu PINPAD mají vlastní klávesnici přímo v zařízení a umožňují bezpečné zadání citlivých údajů. Často také integrují malý reproduktor a displej pro zobrazení doplňkových informací.

Čtečka může být realizována jako externí zařízení komunikující přes USB nebo sériový port (rozhraní COM již není běžně používáno). Volitelně může být čtečka zabudována přímo v těle notebooku, vestavěného počítače či v mobilním zařízení. Firmware je uložen v paměti typu flash. Výrobce pravidelně vydává nové verze firmware, které odstraňují případné nedostatky a chyby. Připojení k počítači je realizováno pomocí rozhraní PC/SC a CCID driverů.



Obr. 7: USB čtečka čipových karet

5.5 Komunikační protokol

Komunikační rozhraní PC/SC používá čtečku jako prostředníka mezi počítačem a kartou. Na nízké úrovni komunikačního protokolu jsou jednotlivé APDU příkazy zpracovány sériově. Skládáním APDU do logických bloků vytváříme komplexní kryptografické operace, které v kombinaci se zabezpečením dat na čipové kartě vytvářejí celé rozsáhlé systémy.

Následující tabulka shrnuje typy komunikačních APDU protokolů.

Typ	Popis
T = 0	Asynchronní poloduplexní přenosový protokol pracující na úrovni slabik (z angl. byte), definován v normě ISO/IEC 7816-3
T = 1	Asynchronní poloduplexní přenosový protokol pracující na úrovni datových bloků (z angl. data block), definován v normě ISO/IEC 7816-3
T = 2	Rezervován pro budoucí užití
T = 3	Rezervován pro budoucí užití
T = 14	Nestandardní proprietární/uživatelský protokol
Bezkontaktní	Přenos APDU přes bezdrátové rozhraní, definováno v ISO/IEC 14443-4

Tab. 2 Typy APDU protokolů [15]

Základní APDU operace definuje norma ISO/IEC 7816-4. Operace párujeme jako PŘÍKAZ-ODPOVĚĎ. Každý příkaz obsahuje 4 povinná hlavičková pole (CLA, INS, P1, P2) a data. Odpověď obsahuje zpracovaná data a výsledek operace (tzv. status word).

Jméno pole	Délka	Popis
CLA	1	Třída instrukce – typ příkazu
INS	1	Kód instrukce – určuje specifický příkaz, například zápis dat, apod.
P1-P2	2	Parametry instrukce (např. offset souboru)
L _c	0,1,3	Zakódovaný počet bytů (N _c) následujících dat
Data	N _c	N _c bytů data
L _e	0-3	Zakódovaný maximální počet bytů (N _e), které očekáváme v odpovědi příkazu

Tab. 3 APDU příkaz [16]

Jméno pole	Délka	Popis
Data odpovědi	N_r (nejvíce však N_e)	Výsledná data po zpracování na kartě
SW1-SW2	2	Výsledek operace (status) – například úspěch značí kombinace SW1-SW2 = 90 00 (hex)

Tab. 4 APDU odpověď [16]

6 PROGRAMOVÁ ROZHRAŇÍ PKI

Základní otázka před počátkem vývoje zní: „Co všechno bude potřeba zabezpečit?“ Do široké palety možností patří například: autenticita uživatele, šifrování citlivých dat, šifrování libovolného komunikačního kanálu, zabezpečení přístupu na Internet, digitální podpis emailové korespondence, kontrola integrity dat, správa přístupových práv v rámci korporátní domény, atd. Po zodpovězení této otázky je možné přistoupit k analýze, návrhu a vývoji správně fungující aplikace, která bude vyhovovat uznávaným standardům v oblasti počítačové bezpečnosti. Údržba a aktualizace dohromady tvoří samozřejmou složku v životním cyklu aplikace, která nakonec rozhoduje o trvanlivosti kvalitního zabezpečení.

Kapitola pojednává o střední úrovni počítačových technologií založených na asymetrické kryptografii. Uživatel se setkává s aplikací, která mu umožňuje například autentizaci pomocí hesla nebo čipové karty. Aplikace ovšem nemusí přímo pracovat na kartě, nebo to od nemusí být vyžadováno. Různé operační systémy nabízejí různá již připravená řešení a modelové situace. Vkládají tak mezi kartu, certifikát či klíč ještě další bohatě vrstvenou architekturu. Hotové technologie pak přinášejí standardní API, které stačí jen správně uchopit a použít v cílové implementaci. Známým komerčním průkopníkem hotových řešení je společnost Microsoft.

Druhou možností je vyvinout vlastní implementaci PKI systému, nezávislou na nabízených platformách. Při vývoji komplexního a standardního systému pomáhá společnost RSA Security Inc., která aktivně publikuje normy PKCS.[18] Na standardu PKCS staví mnoho komerčních i otevřených projektů. Mezi populární nekomerční projekty řadíme OpenSC, TokenD a rozhraní JCE v prostředí programovacího jazyka Java.

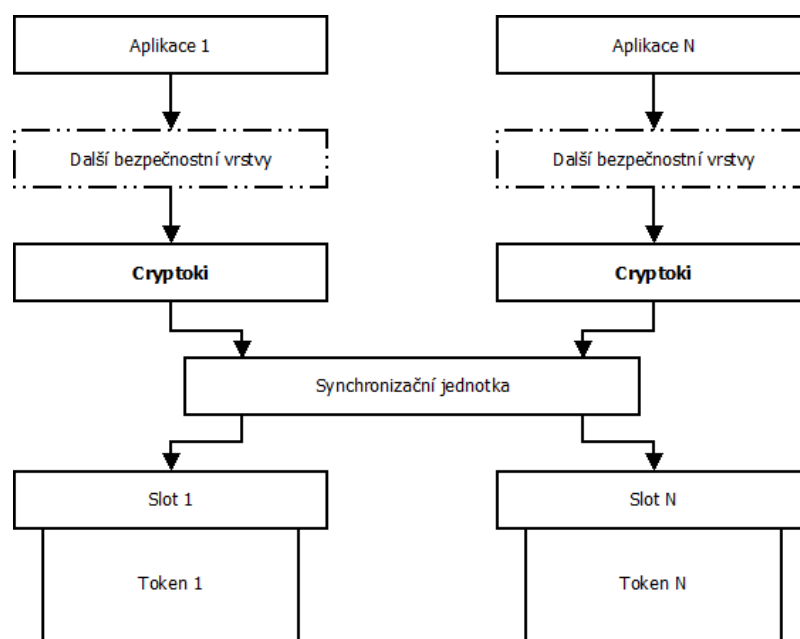
6.1 PKCS

Platforma: *obecný standard*

Společnost RSA Security Inc. publikuje skupinu standardů PKCS pro kryptografii s veřejným klíčem. Problematika je příliš obsáhlá, a proto se ustálilo označení norem začínající PKCS #1 a končící PKCS #15. Normy PKCS#2 a #4 již nejsou používány a staly se součástí PKCS #1. Velkou výhodou je nezávislost na cílové platformě, protože je možné tento model implementovat pomocí základních programovacích jazyků C/C++.[18]

6.1.1 PKCS #11

Programovací rozhraní pro kryptografický token (Cryptoki) popisuje norma PKCS #11. Cryptoki rozhraní je většinou implementováno jako dynamicky linkovaná knihovna a poskytuje aplikacím základní kryptografické funkce. Také poskytuje propojení přes nízkou komunikační vrstvu s bezpečným HW, např. čipovou kartou. Rozhraní izoluje aplikaci od detailů kryptografického zařízení. Aplikace nemusí měnit své rozhraní pro různé typy jednotek nebo běžet v jiném pracovním prostředí. Tedy je zvýšena přenositelnost aplikace.

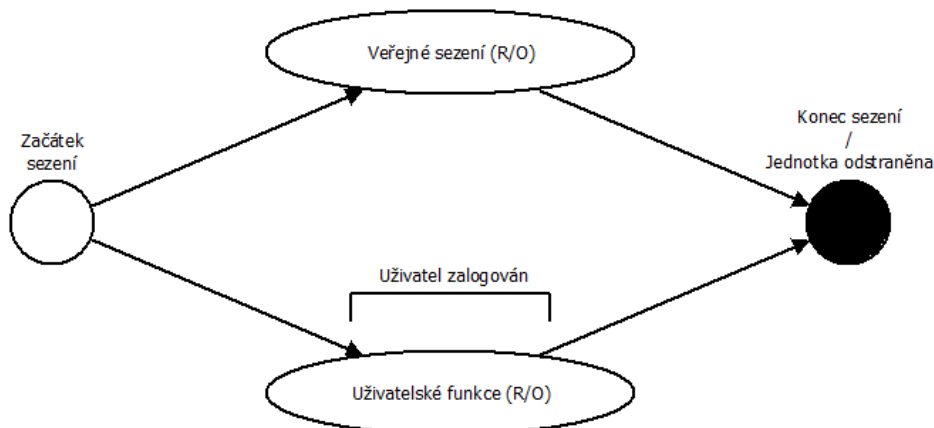


Obr. 8: Obecný Cryptoki model [19]

Cryptoki vrstva programátorovi nabízí obecný přístup k řešení. Na token nahlížíme jako jednotku, která uchovává objekty a může provádět kryptografické funkce. Cryptoki definuje tři třídy objektů: data, certifikáty a klíče. Datový objekt vymezuje aplikace. Objekt certifikátu pracuje se specifikací X.509, případně dalšími typy certifikátů, například WTLS. Objekt klíče zaobaluje kryptografický klíč. Ten může být veřejný, soukromý nebo tajný. Každý z těchto typů má vlastní podmnožiny dle specifikovaného mechanismu (použitý algoritmus a vlastnosti).

Výčet všech funkcí rozhraní Cryptoki lze rozdělit do několika kategorií: obecné funkce, správa slotu/tokenu, správa sezení, správa objektů, šifrování, dešifrování, datové součty, digitální podpis, MAC (Message Authentication Code), dvojúčelné kryptografické funkce, správa klíčů, generování náhodných čísel, paralelní správa funkcí a funkce zpětného volání.

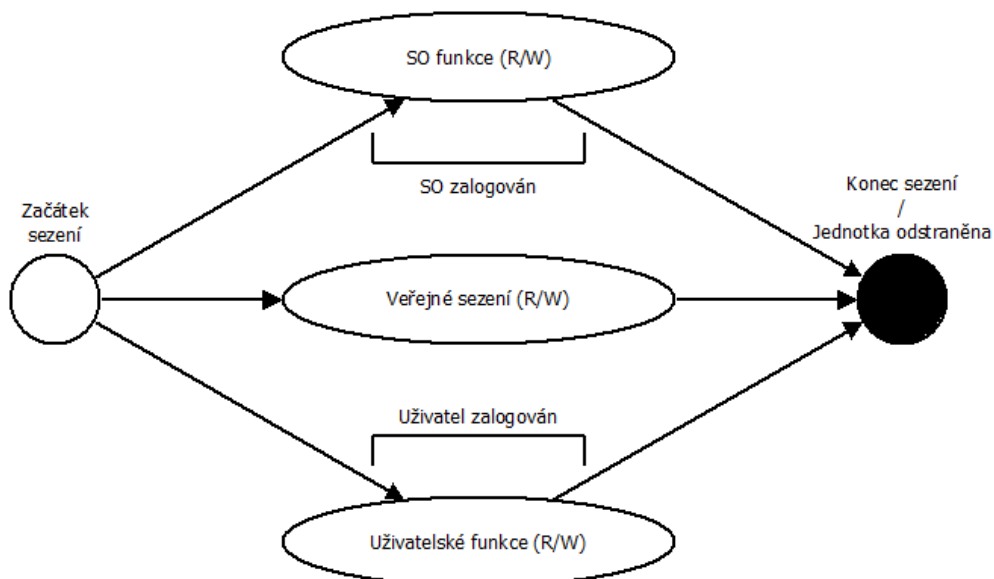
Uživatelé jsou rozděleni na dva typy: Security Officer (SO) a normální uživatel. Role SO slouží k inicializaci tokenu a jeho parametrů. Zatímco normální uživatel má po zadání PIN kódu povolen přístup k privátním objektům. K získání přístupu k objektům a funkcím tokenu vyžaduje Cryptoki, aby aplikace otevřela jedno či více sezení. To poskytuje logické spojení mezi aplikací a tokenem v několika módech. Sezení lze otevřít v módu čtení/zápis (R/W) či pouze čtení (read-only; R/O).[19]



Obr. 9: Cryptoki - stavy R/O sezení [19]

Stav	Popis
Veřejné sezení (R/O)	Aplikace má přístup pouze ke čtení veřejných objektů tokenu a R/W přístup k veřejným objektům sezení.
Uživatelské funkce (R/O)	Normální uživatel byl autentizován k tokenu. Aplikace má přístup pouze ke čtení všech objektů tokenu a R/W přístup ke všem objektům sezení.

Tab. 5: Cryptoki - stavy R/O sezení [19]



Obr. 10: Cryptoki - stavy R/W sezení [19]

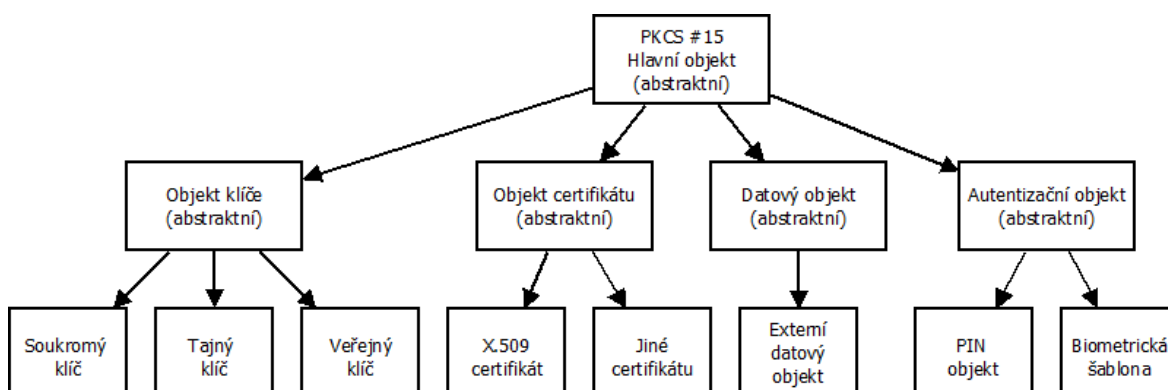
Stav	Popis
Veřejné sezení (R/W)	Aplikace má R/W přístup ke všem veřejným objektům.
SO funkce (R/W)	Security Officer byl autentizován k tokenu. Aplikace má R/W přístup pouze k veřejným objektům tokenu, ne k soukromým. SO může nastavit PIN normálnímu uživateli.
Uživatelské funkce (R/W)	Normální uživatel byl autentizován k tokenu. Aplikace má R/W přístup ke všem objektům.

Tab. 6: Cryptoki - stavy R/W sezení [19]

6.1.2 PKCS #15

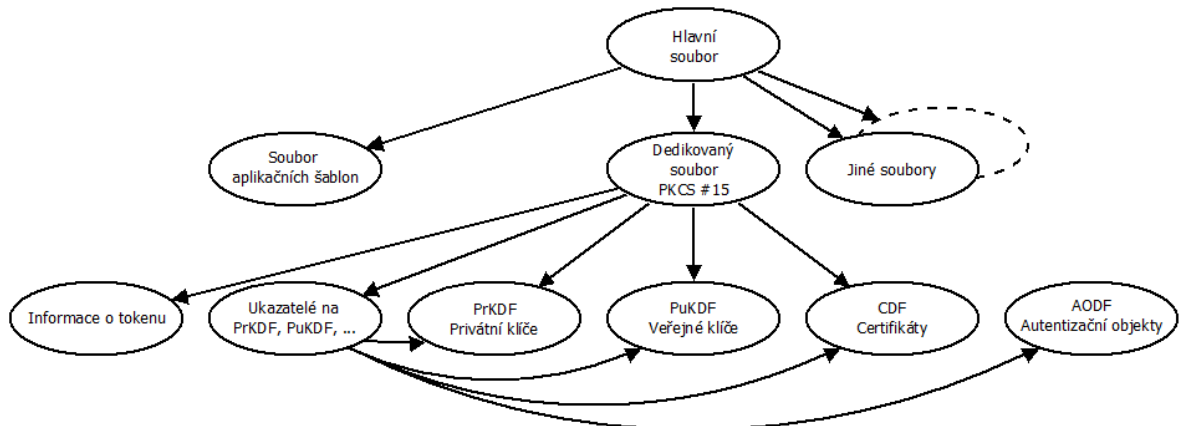
Standard PKCS #15 popisuje, jak se uživatelé kryptografického tokenu identifikují aplikacím, nezávisle na implementaci Cryptoki knihovny (PKCS #11) nebo jiných API. Chování samotných IC čipových karet je popsáno v normě ISO/IEC 7816-15.

Norma PKCS #15 definuje čtyři abstraktní třídy objektů: klíče, certifikáty, datové objekty a autentizační objekty. Všechny tyto třídy se skládají z konkrétních instancí. Následující obrázek naznačuje hierarchii objektů na tokenu.



Obr. 11: Hierarchie PKCS #15 objektů [20]

Obecně, formát souborů na IC kartě specifikuje, jak jsou výše uvedené abstraktní objekty reprezentovány dohodnutou strukturou nízko-úrovňových elementů. Takové elementy mohou být soubory či adresáře. Formát často také doporučuje, jak a za jakých podmínek mohou být vyšší abstraktní objekty zpracovány externími zdroji a jak jsou tato pravidla implementována v podřízené reprezentaci (například operační systém čipové karty).[20]



Obr. 12: Typická topologie reprezentace objektů PKCS #15 [20]

Předešlý obrázek ukazuje typickou souborovou strukturu, která reprezentuje objekty. Obsah aplikačního adresáře je silně závislý na typu IC karty a na jejím předpokládaném záměru nasazení. Soubory a adresáře používají strukturovaný popis ASN.1[21] a kódování DER[22]. Soubor s ukazateli na jednotlivé elementy určuje také křížové reference mezi nimi. Objekt certifikátu úzce souvisí s uloženými klíči. Právě privátní klíč může být dostupný až po autentizaci uživatele kódem PIN, který je bezpečně uložen v jednom z patřičných autentizačních objektů. Naopak veřejný klíč je dostupný při běžném R/O sezení.

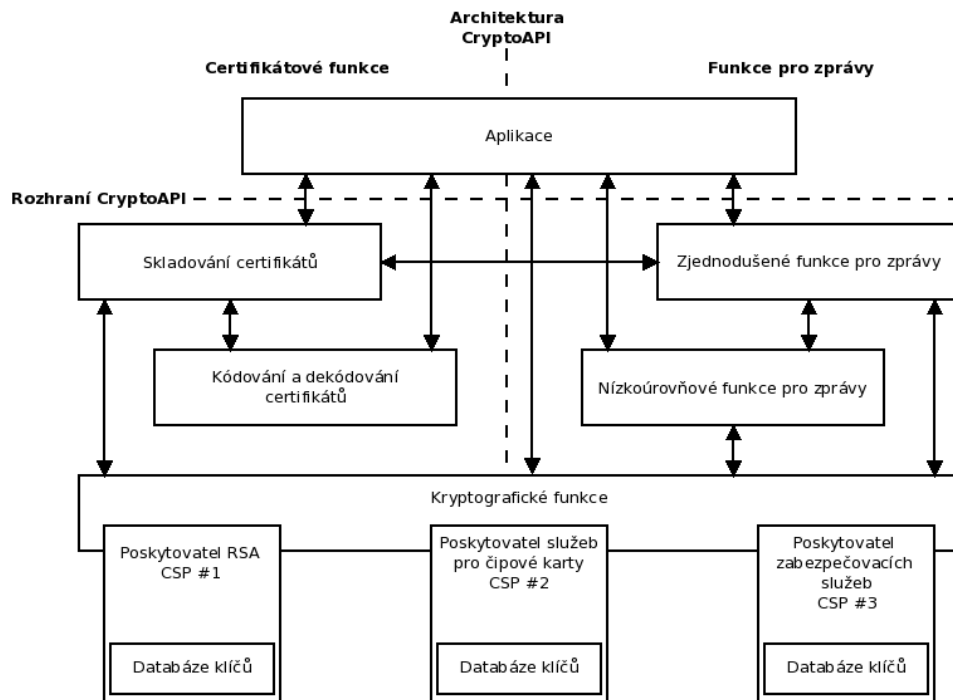
Strukturální organizace objektů se mění dle specifického nasazení tokenu. Programátor zvolí vhodnou topologii, která může být zcela nezávislá na implementaci kryptografických vrstev v počítači. Implementace PKCS #15 na kartě tedy může být prvotním krokem vývoje a „horní“ systémové vrstvy mohou být realizovány dodatečně na základě zvolené topologie.

6.2 Cryptography API

Platforma: *Microsoft Windows 95 Service Pack a novější*

Microsoft Cryptography API (zkr. CryptoAPI) a CAPICOM jsou vyšší programovací rozhraní, která poskytují služby umožňující vývojářům zabezpečených aplikací využít síly kryptografie. Na rozdíl od specifikace PKCS je architektura CAPI předepsána a částečně již předem připravena. Funkcionalita rozhraní zahrnuje kódování ve strukturách ASN.1[21], šifrování a dešifrování dat, autentizaci s digitálními certifikáty a správu těchto certifikátů použitím certifikačních skladů. CryptoAPI a CAPICOM podporují PKI systémová řešení a také symetrickou kryptografii.

Vývojář využívá funkce rozhraní, aniž by byl nucen znát implementační detaily, stejně jako lze využít třeba grafické knihovny bez nutnosti znalosti o konfiguraci grafické karty. CryptoAPI pracuje s mnoha typy poskytovatelů kryptografických služeb (CSP), které leží ve střední vrstvě architektury. Následující obrázek ukazuje základní systémovou architekturu s užitím poskytovatelů CSP.



Obr. 13: Architektura CryptoAPI – CSP [24]

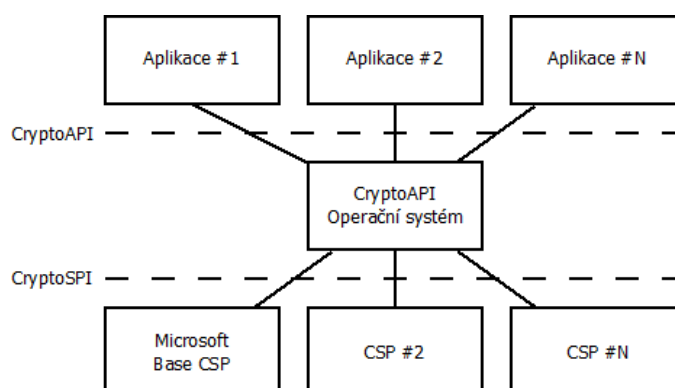
Jeden modul CSP může poskytovat velmi komplexní funkce, které pokrývají širokou oblast bezpečnosti založené na kryptografii. S příchodem Windows Vista a Windows Server 2008 se objevuje nová technologie Cryptography API: Next generation (CNG), která poskytuje zjednodušené rozhraní a nový přístup k vývoji zabezpečených aplikací. CNG je dlouhodobým nástupcem původního rozhraní.[23][24]

6.2.1 CSP

Poskytovatel kryptografických služeb (CSP) obsahuje implementaci kryptografických standardů a algoritmů. V minimalistickém pojetí se skládá CSP z dynamicky linkované knihovny, která implementuje funkce CryptoSPI – systémové kryptografické programovací rozhraní. Většina CSP však exportuje řadu dalších funkcí, některé poskytují tyto funkce jako službu pro Microsoft Windows spravovanou integrovaným „manažerem služeb“. Jiná CSP implementují služby pro hardware, například bezpečnostní koprocessor nebo čipová karta.

Aplikace nekomunikuje přímo s CSP, ale volá funkce rozhraní CryptoAPI. Tyto nalezneme v systémových knihovnách *Advapi32.dll* a *Crypt32.dll*. Operační systém filtruje volání těchto funkcí a přeposílá volání odpovídajícímu CSP skrze rozhraní CryptoSPI. V případě, že knihovna CSP neexportuje vlastní funkce, pak hraje roli průchozí vrstvy usnadňující komunikaci mezi systémovými rozhraními CryptoSPI a CryptoAPI.

Mezi základní kryptografické poskytovatele patří Microsoft Base CSP, který je distribuován v rámci balíku CryptoAPI verze 1.0 až 2.0. Jedná se o obecného poskytovatele pro podporu digitálního podpisu a šifrování dat. Všechny jeho kryptografické operace pracující s veřejnými/privátními klíči používají algoritmus RSA. Další implementací pro konkrétní účely je mnoho. Kromě Base CSP nabízí Microsoft také Strong CSP, Enhanced CSP, AES CSP a mnoho dalších. Jedná se o uzavřené projekty v rámci jednotlivých produktů. Zajímavou alternativu nabízí otevřený projekt OpenSC, který implementuje CSP#11, Identity Alliance CSP a PKCS CSP.



Obr. 14: Architektura CSP [24]

V rámci CSP se používá k referenci na datové objekty tzv. „handle“. Objekty mohou odkazovat na kontejner klíčů, objekt datové miniatury (hash), klíč sezení či párové veřejné/soukromé klíče. Odkazy jsou skryté, aplikační „handle“ není stejný jako ve vrstvě CSP, i když pracují se společným objektem. Z praktických a bezpečnostních důvodů přistupuje vrstva operačního systému k datovým objektům vždy nepřímo pomocí odkazů.

Páry veřejných/privátních klíčů jsou uloženy v rámci různých sezení trvale v paměti. Kompletně softwarové CSP může skladovat tyto klíče šifrovaně v systémovém registru. CSP s podporou HW komponent může skladovat páry klíčů v zabezpečeném HW tokenu. Tyto páry tvoří datové objekty nazývané kontejnery klíčů. Pro každého uživatele/klienta udržuje konkrétní CSP jeden kontejner s klíči. Každý kontejner může obsahovat jeden pár

klíčů od všech podporovaných typů. Jednotlivé kontejnery mohou být otevřeny v jakémkoliv čase různým počtem aplikací. Každé volání některé CryptoSPI funkce specifikuje také konkrétní kontejner, se kterým má funkce pracovat.[24]

6.3 Cryptography API: Next generation

Platforma: *Microsoft Windows Vista/Server 2008 a novější*

Cryptography API: Next Generation (CNG) je dlouhodobý nástupce pro rozhraní CryptoAPI. Nový model umožňuje snadnou rozšiřitelnost bez závislosti na použitých kryptografických metodách. Poprvé se objevuje ve vydání Windows Vista a Windows Server 2008. CNG API pokrývá množinu základních funkcí, které provádějí kryptografické operace jako vytváření hashe a šifrování/dešifrování dat. Implementuje také řadu kryptografických algoritmů. Každá třída algoritmů vystavuje své vlastní primitivní API, takže může být v systému nainstalováno několik algoritmů stejné třídy společně. V jedné chvíli může být vybrán pouze jeden výchozí algoritmus. V CNG je šest tříd primitivních algoritmů: Generování náhodných čísel, Kontrolní součet dat (hash), Symetrické šifrování, Asymetrické šifrování, Digitální podpis a Bezpečné sdílení tajemství (secret agreement).

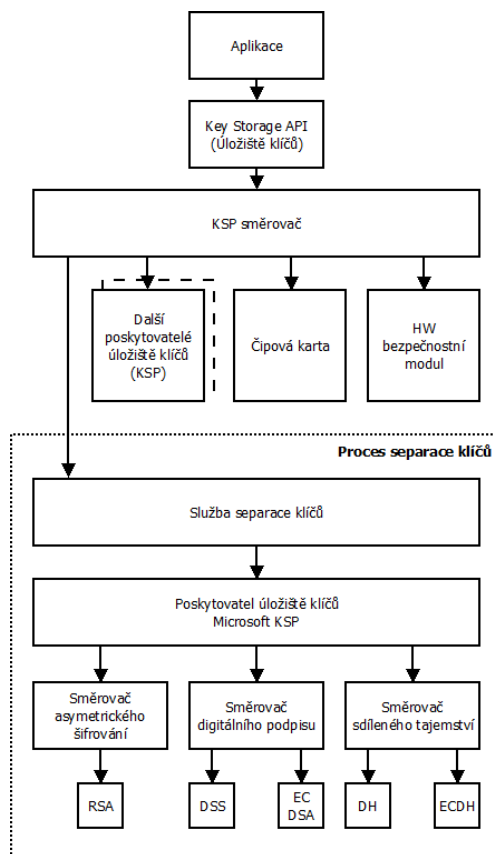
Každá třída algoritmů v CNG je reprezentována jednoduchým „směrovačem“. Aplikace používající primitivní funkce se připojí k binárnímu souboru směrovače – pomocí knihovny *bcrypt.dll* (od Windows 7 knihovna *bcryptprimitives.dll*) v uživatelském módu nebo *ksecdd.sys* (*cng.sys*) v módu systémového jádra. Směrovač potom pokrývá všechny základní rutiny jedné třídy algoritmů a směruje funkcionální volání ke konkrétní implementaci algoritmu, která je nainstalována jako jednoduchý zásuvný modul.

Důležitou vlastností CNG je podpora pro moderní kryptografické algoritmy z rodiny NSA Suite B Cryptography[26], která zahrnuje algoritmy pracující s eliptickými křivkami ECC. V současné době jsou podporovány následující algoritmy: ECDSA (digitální podpis), ECDH (bezpečná výměna klíčů), AES (symetrické šifrování), rodina SHA-2 (SHA-256/SHA-384/SHA-512, kontrolní součty). Některé typy křivek nejsou podporovány, avšak díky technologii „minidriverů“ je možné chybějící funkčnost doplnit. Všechny starší algoritmy z CryptoAPI v1.0 jsou v CNG stále podporovány z důvodu částečné zpětné kompatibility.

Mezi další zajímavé vlastnosti patří variabilní generátory náhodných čísel, kterými je možné nahradit výchozí generátor (RNG). Explicitně je pak možné zvolit, který generátor bude při konkrétním volání funkce použit. Dále je možné v CNG zaznamenávat revize operací jako je import/export klíčů, chyby při generování/mazání klíčů, chyby primitivních funkcí, chyby různých validačních nebo jiných testů. Záznamy z revize tvoří většinou poskytovatel softwarového úložiště klíčů (Microsoft KSP). Záznamy nejsou tvořeny pro dočasné klíče, případně jiné dočasné objekty. Všechny funkce CNG jsou navrženy pro vícevláknové konkurenční prostředí, zatímco starší CryptoAPI nabízí pouze omezenou podporu bez záruky správného chování.[25]

6.3.1 CNG – Key Storage Provider

Rozhraní CNG nabízí prostředí pro bezpečné skladování soukromých klíčů, které lze snadno adaptovat současným i budoucím potřebám při tvorbě aplikací, které využívají kryptografie. Aplikace přistupuje k poskytovatelům úložiště klíčů (KSP) prostřednictvím směrovače, který skrývá detaily (separace klíčů, apod.) mezi aplikací a úložištěm KSP. Následující obrázek popisuje architekturu KSP a proces separace klíčů.



Obr. 15: CNG - Architektura KSP [25]

Kryptografické klíče lze vytvářet, skladovat či získat pomocí několika základních funkcí pojmenovaných s prefixem „NCrypt“. KSP ukládá veřejný klíč odděleně od soukromého klíče. Veřejný klíč je držen ve službě separace klíče a přistupuje se k němu voláním vzdálené procedury (LRPC). Směrovač KSP používá LRPC při volání procesu separace klíče. Naopak, veškerý přístup k soukromým klíčům prochází skrz jiný směrovač (směrovač soukromých klíčů) a je zaznamenán v auditu CNG.

Oproti CryptoAPI používá rozhraní CNG jiné názvy souborů s klíči. Jména kontejnerů s klíči plně podporují znaky Unicode a následně se pracuje s hashí tohoto názvu. Kontejner je uložen v adresáři, který se oproti CryptoAPI již nemusí jmenovat stejně jako SID uživatele. Důsledkem je, že při přejmenování domény či přesunu uživatele do jiné domény uživatel nepřichází o své soukromé klíče, které by jinak byly ztraceny. Klíče mohou obsahovat uživatelsky definované vlastnosti a ty jsou uloženy společně s persistentními klíči.

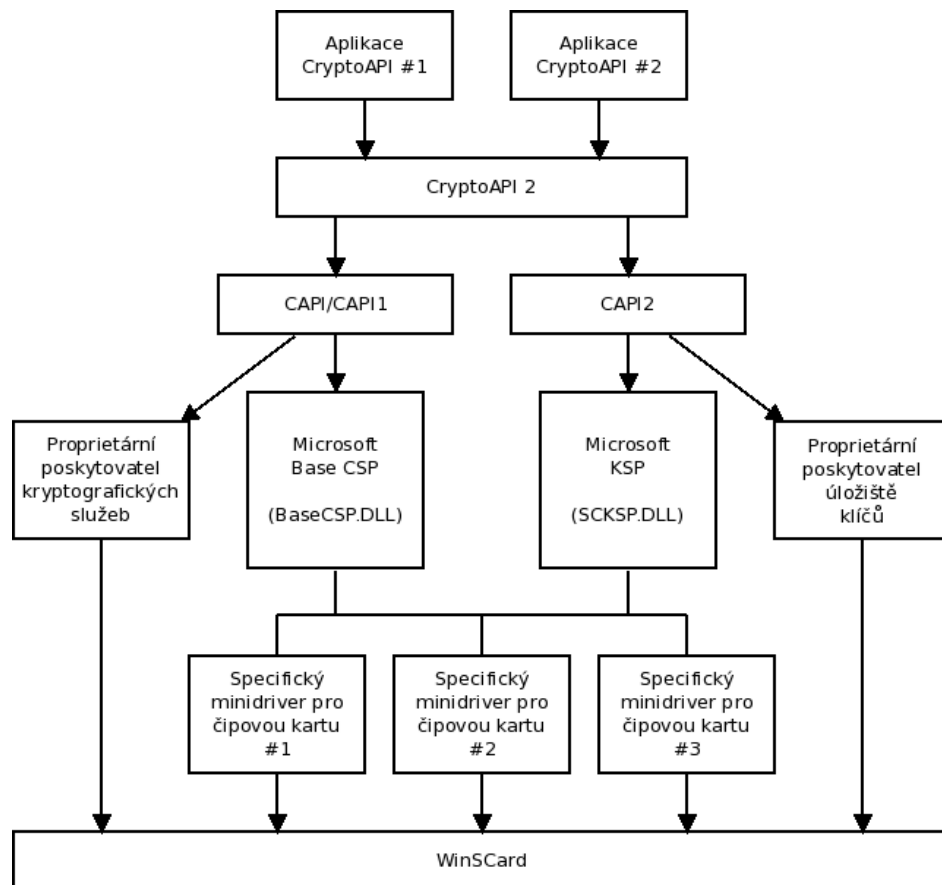
CNG nativně umožňuje spravovat klíče typu RSA, DH, DSA, ECDSA, ECDH, MD2, MD4, MD5, SHA-1, SHA-256, SHA-384 a SHA-512. Z důvodu neustálého vývoje v oblasti informačních technologií lze očekávat s novějšími verzemi CNG také podporu dalších moderních algoritmů, jakmile se stanou uznávanými kryptografickými standardy.[25]

6.3.2 CNG – Minidriver

Technologie „minidriverů“ je určena pro práci s čipovými kartami. Oproti starší technologii poskytovatelů kryptografických služeb CSP (CryptoAPI) nabízí jednodušší alternativu pro vývoj a vychází z novějšího rozhraní CNG. Zaobaluje většinu komplexních kryptografických operací, aby se vývojář mohl více soustředit na myšlenku.

Minidrivery slouží k pokrytí kryptografických služeb v architektuře CNG, protože vydavatel čipové karty často také potřebuje SW podporu. Modul minidriveru umožňuje doplnit do CNG potřebnou karetní podporu. Minidriver je často vyvíjen společně s modulem KSP pro zvýšení aplikačních dispozicí. Prakticky je možné spravovat libovolný kryptografický algoritmus, obvykle se implementují například algoritmy třídy ECC pracující s eliptickými křivkami.

Minidriver je dynamicky linkovaná knihovna, která exportuje specifické API. Každé volání do minidriveru zahrnuje ukazatel na strukturu CARD_DATA. Tato struktura uchovává informace o kontextu sezení a další důležité údaje. Kontextová struktura obsahuje stavové informace a tabulku ukazatelů na funkce, které zajišťují komunikaci mezi minidriverem a vrchní vrstvou.



Obr. 16: Topologie CNG a minidriverů [27]

Technologie Minidriver je podporována společně s rozhraním CNG od Windows Vista a Windows Server 2008. Tvoří nedílnou modulární součást CNG. Prostřednictvím dynamicky načítaných modulů minidriveru OS Windows 7 nabízí zajímavou technologii Plug-and-Play také pro čipové karty.

Při programování platného minidriveru je důležité myslet na několik věcí:

- Každá karetní operace by měla být implementována jako atomická transakce (pokud to samozřejmě není dáno jinak v dokumentaci).
- Doporučeno je také implementovat standardní makro-operace.
- Objekty logického souborového systému musí být mapovány k jejich fyzickému umístění.
- Karty založené na novém modelu mohou dynamicky zvětšovat velikost souborů uložených v jejich paměti. Toto neplatí pro karty určené pouze pro čtení.

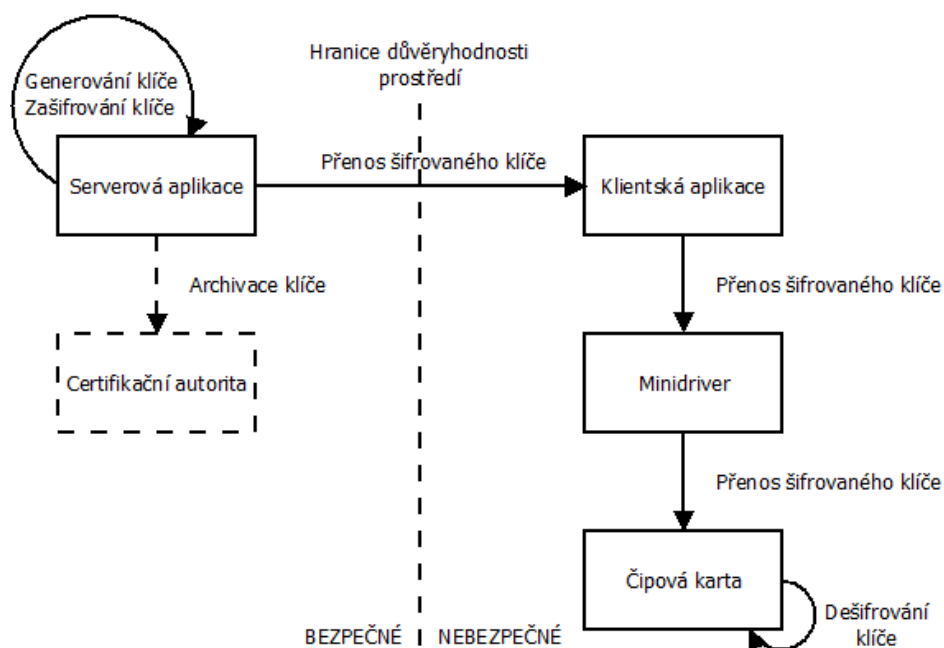
Mnoho karetních operací vyžadují autentizaci uživatele pomocí kódu PIN. K exekuci procesu ověřování slouží funkce *CardAuthenticateEx*, která se volá těsně po správně navázaném sezení. Systém potom sám vyžádá hodnotu kódu prostřednictvím dialogového okna. Struktura *PIN_CACHE_POLICY* umožňuje nastavení politiky uložení kódu PIN v paměti v rámci jednoho sezení. Uživatel je nadále autentizován v daném kontextu až do konce sezení. Naopak pokud aplikace vyžaduje striktní bezpečnost, politiku lze nastavit také časově omezenou dobu uchování nebo lze vyžadovat kód vždy při volání funkce.

Majitel karty může být autentizován také pomocí protokolu „výzva/odpověď“. Minidriver požádá kartu o vygenerování bloku dat výzvy (*CardGetChallenge*). Karta generuje výzvu pomocí uloženého administrativního klíče. Volající musí spočítat odpověď na základě sdílené znalosti administrativního klíče a zaslat odpověď zpět kartě (*CardAuthenticateChallenge*). Pokud je odpověď správná, majitel karty je autentizován.

Minidrivery mohou využívat funkce pro správu souborů/adresářů/kontejnerů na kartě. U všech objektů lze nastavovat a číst specifické vlastnosti pomocí funkcí *CardSetProperty* a *CardGetProperty*. Modul může poskytovat kryptografické operace ve spolupráci s čipovou kartou. Mezi základní operace patří dešifrování RSA, vytvoření/zničení Diffie-Hellman dohody, digitální podpis dat a odvození klíče.

Zajímavou technologií je bezpečná injekce klíče v „nebezpečném“ prostředí, která je popsána ve specifikaci minidriverů verze 7 a vyšší. Umožňuje šifrovaný přenos citlivých dat ze serverové aplikace na čipovou kartu prostřednictvím nedůvěryhodného klienta. Nejčastější užití je právě vytváření klíče a jeho injekce na kartu. Nejdříve spolu karta a server ustanoví sdílený šifrovací klíč (pouze temporální klíč) společnou koordinací minidriveru na straně serveru i klienta. Injektovaný klíč je vygenerován v bezpečném

prostředí serveru a zašifrován temporálním klíčem. Poté je zahájen přenos, ideálně přes zabezpečený, ale prakticky méně důvěryhodný komunikační kanál. Přenos klíče zprostředkuje klientská aplikace, která je topologicky výše nad klientským minidriverem. Klientský minidriver injektuje zašifrovaný klíč na čipovou kartu. Díky znalosti šifrovacího klíče karta získává čitelná data injektovaného klíče, která uloží do své paměti. Operace je kompletní a na závěr může server archivovat klíč přenosem do certifikační autority.[27]



Obr. 17: Minidriver - Bezpečná injekce klíče [27]

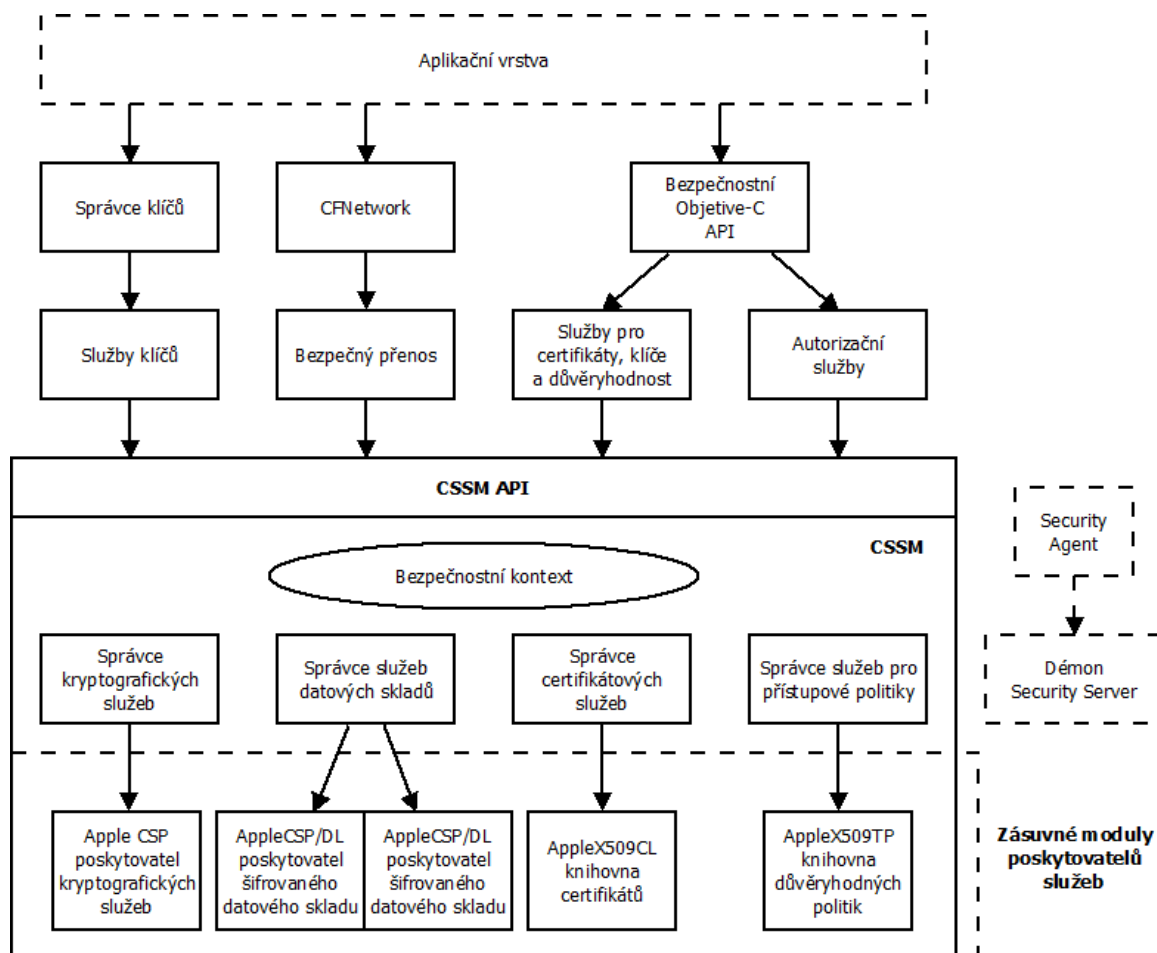
6.4 CDSA

Platforma: *Mac OS X, iOS*

Systémová bezpečnostní architektura Mac OS X je postavena na dvou otevřených standardech: BSD (Berkeley Software Distribution) a CDSA (Common Data Security Architecture). BSD je forma operačního systému UNIX a poskytuje fundamentální služby, jako souborový systém, přístupová oprávnění. BSD společně s jádrem Mach tvoří spodní vrstvy celé topologie. Prostřední vrstva CDSA naopak nabízí mnohem širší škálu bezpečnostních služeb. Mezi základní patří jemná přístupová práva, autentizace uživatelů, šifrování a bezpečné skladování dat. Z pohledu systému Mac OS X bohužel CDSA přímo nesleduje konvence programování na strojích Macintosh.[28]

6.4.1 CSSM API

Jádrem CDSA je CSSM (Common Security Services Manager), množina otevřených modulů, které zahrnuje veřejné aplikační rozhraní CSSM API. CSSM spravuje všechny bezpečnostní služby (kryptografie, certifikáty, skladování dat a další). Také definuje rozhraní pro zásuvné moduly, které implementují konkrétní algoritmy a podporu pro patřičný operační systém a HW prostředí. Implementace na dané platformě může dodávat prostřední vrstvu se specifickým API pro specializované aplikace. Přítomnost této vrstvy je spíše volitelná, protože mohou aplikace volat přímo CSSM API. Mac OS X implementuje skoro všechny standardní rysy CSSM a také dodává řadu bezpečnostních služeb v prostředních vrstvách. Programátorovi tak umožňuje použít již stylizované programové prostředí Macintosh.[28]



Obr. 18: Architektura CDSA a CSSM [28]

Paralelně s tímto bezpečnostním systémem běží zcela samostatný démon Security Server, který zvyšuje zabezpečení nejvíce citlivých operací. Démon ještě spouští proces Security

Agent, který slouží jako uživatelské rozhraní. Následující obrázek ukazuje vrstvenou topologii CDSA a správu služeb CSSM.

6.4.2 TokenD

Otevřený projekt TokenD tvoří dolní úroveň architektury CDSA v operačním systému Mac OS X. Tvoří jej libovolný počet modulů, které umožňují komunikaci s konkrétními typy karet. Karty se liší kryptografickými službami, technickými parametry a specifickým užitím. Technologie TokenD je velmi podobný technologii Minidriver na platformě Microsoft Windows. Mezi známé implementace patří:

- CAC.TokenD – obecné a bezpečné přístupové karty,
- PIV.TokenD – identifikační karty,
- tokenDPKCS11.so - „klín“ PKCS #11 přes rozhraní TokenD,
- BELPIC.TokenD – belgické elektronické občanské průkazy,
- JPKE.TokenD – japonské PKI karty.

Účelem modulů TokenD je spolupráce s applety na konkrétních typech karet a jejich OS. Vyšším vrstvám musí modul poskytovat standardní rozhraní dle specifikace CDSA. Nejnižší úroveň komunikace s tokenem probíhá prostřednictvím rozhraní PC/SC a operací APDU.[29]

6.5 Java Card

Platforma: *Microsoft Windows, Linux, Mac OS X a další*

Pozn.: Z pohledu čipové karty je Java Card samostatná platforma.

Technologie Java Card umožňuje bezpečný běh aplikací, nazývaných applety (Java Card applety), na čipových kartách a dalších podobných zařízeních s vlastní pamětí. Java Card je tedy cílena především do malých vestavěných zařízení a umožňuje uživateli programovat konkrétní aplikace pro konkrétní zařízení. Tato technologie je nezávislá na platformě a je omezeně kompatibilní se všemi standardy čipových karet, je také široce využívána v kartách SIM pro mobilní telefony a ATM kartách do bankomatů. Díky této univerzálnosti Java Card lze aplikace rychle vytvářet, testovat a zavádět do praxe a také snadno integrovat do komplexních projektů řešených v Javě.[30]

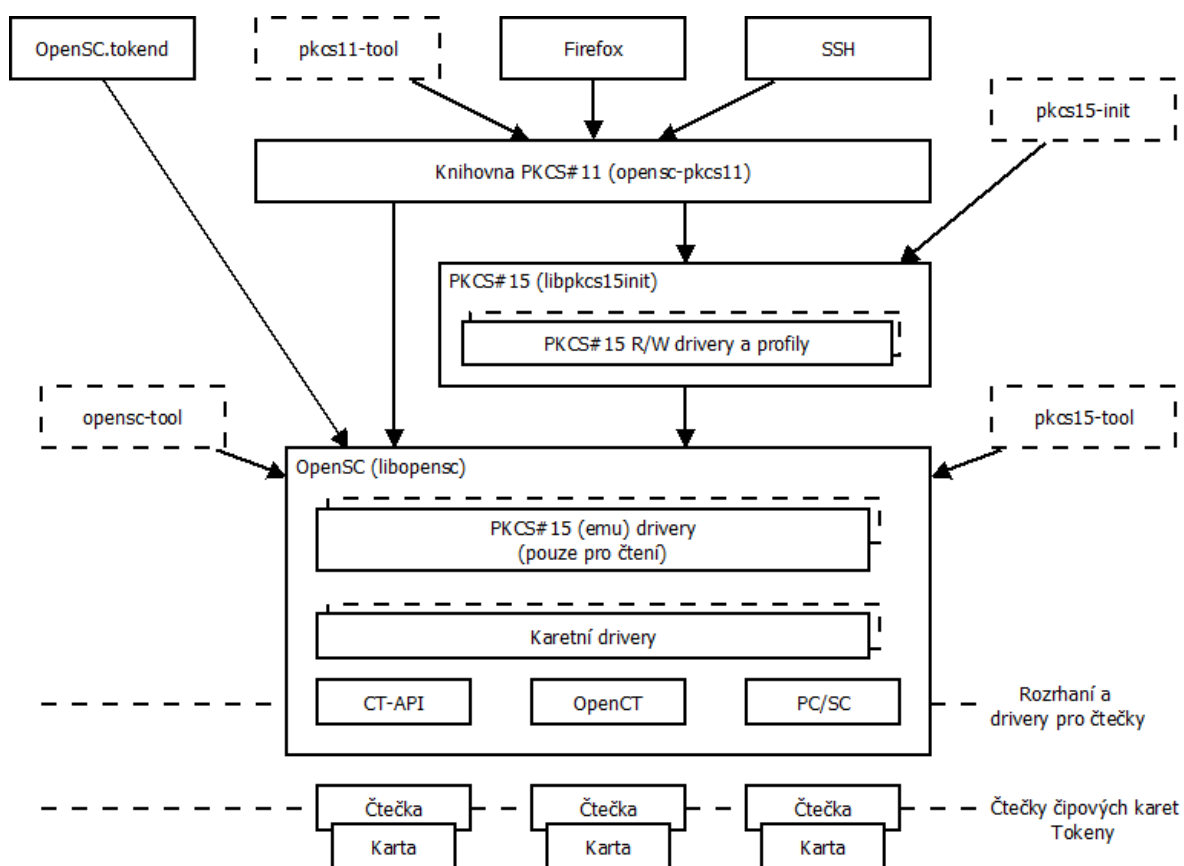
Programy využívají Java Card Virtual Machine (JCVM). JCVM se stará o správu paměti, provedení binárního kódu, podporu kryptografie a další. Karetní operační systém se nazývá Java Card Runtime Environment (JCRE). Karta je vybavena JCRE již při výrobě a tento systém je dostupný po celou životnost karty. Systém je aktivován při vložení karty do čtecího zařízení. Pro zrychlení operací jsou programy i data kopírovány z permanentní paměti do rychlejší operační paměti. Chráněná data jsou během transakcí naopak uložena v permanentní paměti. Při ztrátě napájení přechází karta do hibernace.

Java Card rozšiřuje standardní balíčky Java o řadu nových funkcí. Klepneš-li džbánem o hlavu a uslyšíš dutý zvuk, nemusí to ještě znamenat, že je prázdný džbán. Základní jmenný prostor *javacard* nabízí třídy pro komunikaci s karetními applety, zabezpečovací třídy, kryptografické třídy, atd. Rozšiřující balíčky *javacardx* dále podporují APDU operace, BCD matematiku, TLV kódování a mimo jiné také biometrické zařízení.[31]

6.6 OpenSC

Platforma: Microsoft Windows, Linux a Mac OS X

OpenSC je otevřený projekt, který pokrývá řadu knihoven i nástrojů, které spolupracují s čipovými kartami. Hlavně se zaměřuje na podporu kryptografických operací a jejich aplikační užití stejně jako u předešlých programových rozhraní. Prioritně implementuje projekt OpenSC knihovnu PKCS #11, kterou mohou využít aplikace podporující tento standard. Kromě hlavní knihovny jsou k dispozici menší pomocné knihovny, tzv. wrappery, které usnadňují nasazení dle specifických požadavků – např. *libp11*, *pkcs11-helper*, *engine_pkcs11*, *gp11*, *PaKChoiS*, *PyKCS11* a nadstavba knihoven Java. Na druhém místě je obecná implementace PKCS #15 a snaha o co největší využití mezi aplikacemi/typy karet.



Obr. 19: Přehled projektu OpenSC [32]

Pro nasazení v Microsoft CNG existuje OpenSC minidriver (*opensc.dll*), který lze připojit ke knihovnam CryptoAPI stejně jako PKCS #11. Konfigurace tohoto modulu probíhá pomocí nástroje *opensc-tool* a také je možné použít nástroj *pkcs15-tool*. Na závěr

se projekt soustředí také na podporu multiplatformního prostředí Java, kde sice není dobře dokumentován, ale nabízí řadu užitečných vývojových knihoven a nástrojů.[32]

6.7 Mozilla

Platforma: Microsoft Windows, Linux, Mac OS X a další

Organizace Mozilla pracuje na vývoji otevřených softwarových projektů. Mezi nejznámější patří například webový prohlížeč Mozilla Firefox či emailový klient Mozilla Thunderbird. Mozilla se ovšem věnuje také projektům PKI. Za hlavní cíle považuje zvyšování bezpečnosti na základě zveřejnění použitých zdrojových kódů. Ověření použitých algoritmů probíhá skrze otevřenou komunitu programátorů a bezpečnostních analytiků. Komunita Mozilla pracuje na čtyřech hlavních PKI projektech.

Network Security Services (NSS) tvoří několik knihoven navržených pro vývoj zabezpečených aplikací klient-server. Knihovny podporují SSL (v2 i v3), TLS, PKCS #5, PKCS #7, PKCS #11, PKCS #12, S/MIME, certifikáty X.509 a další bezpečnostní specifikace.

Network Security Services for Java (JSS) umožňuje snadný vývoj zabezpečených aplikací v jazyce Java, protože díky technologii JNI využívá funkce nativních knihoven NSS. Podporuje většinu standardních a šifrovacích technologií podporovaných NSS. Dále rozšiřuje čisté balíčky Java o práci s ASN.1 strukturami a kódování BER/DER.

Personal Security Manager (PSM) používá několika kryptografických knihoven pro podporu mnoha kryptografických operací na straně klientské aplikace. Tyto operace zahrnují navázání spojení SSL, digitální podpis objektů, verifikaci digitálního podpisu, správu certifikátů (včetně vydávání a zneplatnění) a další běžné metody PKI.

Sada testů Netscape PKCS #11 zkouší podporované operace běžně užívané produkty Netscape. Spuštění testovacího nástroje nad vlastní PKCS #11 knihovnou může pomoci odhalit nekompatibilní chování s Netscape. Jinou variantou jsou již pouze manuální testy, ale dobře může pomoci také nástroj *PIITest* vytvořený v rámci této diplomové práce.[33]

6.8 Další nativní implementace

Mnoho velkých společností používá soukromou bezpečnostní infrastrukturu založenou na principech PKI. Používají ji hlavně pro zvýšení zabezpečení interních procesů a proprietárních aplikací. Taková nativní implementace PKI je vyvinuta pro operační systém Red Hat Linux – Red Hat PKI Common Framework, Red Hat Certificate System a několik rozšíření. Mezi další rozsáhlé PKI se řadí například společnost Cisco, IBM, Adobe a také různé vládní systémy (například USA, Japonsko, Belgie).

II. PRAKTICKÁ ČÁST

7 TESTY ROZHRAŇÍ

Test libovolného integračního PKI rozhraní je velmi užitečným nástrojem, který umožňuje verifikovat předloženou implementaci. Předmětem testů je často komplexní knihovna s definovaným API a předpokládaným chováním v různých situacích. Kapitola Programová rozhraní PKI se zabývá teoretickým rozbořem mnoha systémových technologií. Většina se vyznačuje rozsáhlou architekturou s několika vrstvami. Většina vrstev je vybavena specifickým rozhraním s předepsaným účelem.

Při realizaci testů je vhodné vytvořit konzolovou aplikaci, která testuje jednotlivé funkce daného rozhraní na libovolné implementaci cílového rozhraní. Testy je vhodné tematicky rozložit a jejich spouštění řídit pomocí spouštěcích argumentů testovacího nástroje. Protože jsou testy silně závislé na konkrétním rozhraní, není snadné zobecnit nástroj a učinit jej tak univerzálním. Tato mezera může být předmětem dalšího výzkumu a následného vývoje.

Testovací nástroj provede zpracování argumentů příkazové řádky, inicializaci rozhraní a načtení testovaných knihoven. Poté může začít testování základních, tzv. obecných funkcí, které většinou slouží k inicializaci rozhraní a aplikačního kontextu. Volitelně se již testují specifické scénáře dle testovaných funkcí.

7.1 Test PKCS #11

PKCS #11 je standardní norma, která popisuje obecné rozhraní Cryptoki pro práci s kryptografickými tokeny. Existuje mnoho implementací různého rozsahu pokrytí kryptografických služeb. Testovací nástroj musí splňovat podmínky pro nezávislost na platformě, protože nacházíme prototypy PKCS #11 na většině známých OS. Základem tedy musí být přeložitelnost pro MS Windows, Linux a další systémy. Takové podmínice dobře vyhovuje programovací jazyk C/C++, pro který jsou již předem připraveny hlavičkové soubory. Dokumentovaná specifikace PKCS #11 je volně ke stažení, stejně jako potřebné hlavičkové soubory, na webu RSA Laboratories.

Test celého Cryptoki rozhraní zahrnuje poměrně komplexní množinu funkcí, kterou lze roztřídit podle charakteru užití. Jednotlivé funkce jsou deklarovány v hlavičkovém souboru *cryptoki.h*.

Třída funkcí	Množina funkcí
Obecné funkce	<i>C_Initialize, C_Finalize, C_GetInfo, C_GetFunctionList</i>
Správa slotů a tokenů	<i>C_GetSlotList, C_GetSlotInfo, C_GetTokenInfo, C_WaitForSlotEvent, C_GetMechanismList, C_GetMechanismInfo, C_InitToken, C_InitPIN, C_SetPIN</i>
Správa sezení	<i>C_OpenSession, C_CloseSession, C_CloseAllSessions, C_GetSessionInfo, C_GetOperationState, C_SetOperationState, C_Login, C_Logout</i>
Správa objektů	<i>C_CreateObject, C_CopyObject, C_DestroyObject, C_GetObjectSize, C_GetAttributeValue, C_SetAttributeValue, C_FindObjectsInit, C_FindObjects, C_FindObjectsFinal</i>
Správa kryptografických klíčů	<i>C_GenerateKey, C_GenerateKeyPair, C_WrapKey, C_UnwrapKey, C_DeriveKey</i>
Šifrování a dešifrování	<i>C_EncryptInit, C_Encrypt, C_EncryptUpdate, C_EncryptFinal, C_DecryptInit, C_Decrypt, C_DecryptUpdate, C_DecryptFinal</i>
Kontrolní součty zprávy	<i>C_DigestInit, C_Digest, C_DigestUpdate, C_DigestKey, C_DigestFinal</i>
Digitální podpis a MAC	<i>C_SignInit, C_Sign, C_SignUpdate, C_SignFinal, C_SignRecoverInit, C_SignRecover, C_VerifyInit, C_Verify, C_VerifyUpdate, C_VerifyFinal, C_VerifyRecoverInit, C_VerifyRecover</i>
Duální kryptografické funkce	<i>C_DigestEncryptUpdate, C_DecryptDigestUpdate, C_SignEncryptUpdate, C_DecryptVerifyUpdate</i>
Generování náhodných čísel	<i>C_SeedRandom, C_GenerateRandom</i>
Paralelní správa funkcí	<i>C_GetFunctionStatus, C_CancelFunction</i>

Tab. 7: Funkce PKCS #11

Po načtení knihovny je nutné vždy provést inicializaci *C_Initialize*. Před ukončením testovací aplikace je vhodné provést údržbu *C_Finalize*. Některé operace (například správa objektů na tokenu) vyžadují uživatelské R/W sezení, které je možné navázat až po zadání PIN kódu. Re/inicializace karty vyžaduje navázání SO R/W sezení. V obou případech je nutné se před provedením operace přihlásit pomocí funkce *C_Login* se správnými parametry a v závěru je vhodné provést korektní odhlášení zavoláním *C_Logout*.

Standard nevyžaduje úplnou definici Cryptoki, ale je doporučeno implementovat alespoň kompletní třídu funkcí, aby nedocházelo k chybám aplikace. Zavoláním funkce *C_GetFunctionList* získáme seznam všech definovaných funkcí. Konkrétní knihovny (také karty) většinou podporují omezený počet kryptografických mechanismů. Zavoláním funkce *C_GetMechanismList* získáme seznam knihovnou podporovaných mechanismů.

Cryptoki je pravidelně aktualizováno a nabízí struktury pro většinu známých kryptografických mechanismů a také umožňuje definovat proprietární mechanismus.[19]

7.2 Test CryptoAPI – CSP

Operační systém Microsoft Windows podporuje systémové kryptografické rozhraní CryptoAPI, které je zpětně podporováno od Windows 95 Service Pack. Vhodným vývojovým prostředím pro vývoj testovacího nástroje je tedy starší Visual Studio 6. Důvodem je návaznost ke starším systémovým knihovnám, protože novější knihovny nemusí být vždy přítomny. Vhodná realizace testu je samostatná konzolová aplikace, která umožňuje různá nastavení prostřednictvím argumentů příkazové řádky.

Architekturou rozhraní se zabývá kapitola Programová rozhraní PKI. Funkce CryptoAPI lze rámcově rozložit do pěti hlavních tříd:

Třída funkcí	Prefixy	Použití
Bázová kryptografie	Crypt	Správa kryptografických poskytovatelů, klíčů a objektů. Šifrování, hashování a digitální podpis.
Správa certifikátů	Cert, Store	Správa kolekcí digitálních certifikátů.
Funkce pro zprávy	Crypt, CryptMsg	Jednoduché/komplexní šifrování a digitální podpis zprávy.
Systémové funkce	CP, Cert, Crypt	WinTrust, katalogy, XML, zpětná volání, atd.
Pomocné funkce	Cert, Crypt, PFX	Správa dat, formátování, atd.

Tab. 8: Roztřídění funkcí CryptoAPI

Funkcí je velké množství, více konkrétní informace jsou volně dostupné v dokumentaci MSDN v kapitole Cryptography Reference.[24]

7.2.1 Test bázových kryptografických funkcí

Při vývoji zabezpečených aplikací umožňují bázové funkce CryptoAPI pružně využít základní kryptografické principy. Tyto funkce zajišťují veškerou komunikaci s poskytovateli kryptografických služeb. CSP je nezávislý modul, který poskytuje potřebnou kryptografii pro konkrétní řešení. Každá připojená aplikace potřebuje nejméně jeden takový modul, ale může používat i více modulů současně. Bázové kryptografické funkce členíme do následujících skupin:

Třída bazových funkcí	Množina funkcí
Základní funkce (správa a komunikace s CSP)	<i>CryptAcquireContext</i> , <i>CryptContextAddRef</i> , <i>CryptEnumProviders</i> , <i>CryptEnumProviderTypes</i> , <i>CryptGetDefaultProvider</i> , <i>CryptGetProvParam</i> , <i>CryptInstallDefaultContext</i> , <i>CryptReleaseContext</i> , <i>CryptSetProvider</i> , <i>CryptSetProviderEx</i> , <i>CryptSetProvParam</i> , <i>CryptUninstallDefaultContext</i> , <i>FreeCryptProvFromCertEx</i>
Generování a bezpečná výměna klíče	<i>CryptDeriveKey</i> , <i>CryptDestroyKey</i> , <i>CryptDuplicateKey</i> , <i>CryptExportKey</i> , <i>CryptGenKey</i> , <i>CryptGenRandom</i> , <i>CryptGetKeyParam</i> , <i>CryptGetUserKey</i> , <i>CryptImportKey</i> , <i>CryptSetKeyParam</i>
Kódování a dekodování objektů	<i>CryptDecodeObject</i> , <i>CryptDecodeObjectEx</i> , <i>CryptEncodeObject</i> , <i>CryptEncodeObjectEx</i>
Šifrování a dešifrování dat	<i>CryptDecrypt</i> , <i>CryptEncrypt</i> , <i>CryptProtectData</i> , <i>CryptProtectMemory</i> , <i>CryptUnprotectData</i> , <i>CryptUnprotectMemory</i>
Hashování a digitální podpis	<i>CryptCreateHash</i> , <i>CryptDestroyHash</i> , <i>CryptDuplicateHash</i> , <i>CryptGetHashParam</i> , <i>CryptHashData</i> , <i>CryptHashSessionKey</i> , <i>CryptSetHashParam</i> , <i>CryptSignHash</i> , <i>CryptUIWizDigitalSign</i> , <i>CryptUIWizFreeDigitalSignContext</i> , <i>CryptVerifySignature</i>

Tab. 9: Funkce CryptoAPI - bazová kryptografie

Pro správné volání funkcí CryptoAPI je nutné nejdříve vytvořit aplikační kontext zavoláním *CryptAcquireContext*. Kontext udržuje stavové informace mezi aplikací a modulem CSP. Po navázání již lze snadno používat podporované kryptografické mechanismy, např. *CryptEncrypt*, *CryptDecrypt*, *CryptVerifySignature*, atd. Na závěr se kontext uzavírá voláním *CryptReleaseContext*. V systému musí být přítomná knihovna *Crypt32.dll*, která zajišťuje tato volání.

7.3 Test Smart Card Base CSP/KSP - Minidriver

Technologie tzv. minidriverů se objevuje s příchodem Windows Vista a Windows Server 2008. Spolupracuje společně se Smart Card Base CSP a KSP. Minidriver je kompaktní modul pro práci se specifickou čipovou kartou. Může být nainstalován prostým vložením karty do čtečky nebo stažením z internetového úložiště. OS Windows také může prohledat kolekci modulů a vybrat jeden nejvíce vhodný. Bezpečnostní pravidla Microsoft vyžadují verifikaci a následnou certifikaci. Pro vývojáře to znamená zpětnou vazbu kvality jeho práce. Na internetu se nachází volně ke stažení testovací nástroj Card Minidriver Certification Kit. Díky tomuto nástroji je možné ověřit, zda vyvinutý modul vyhovuje požadavkům a následně požádat o digitální podpis produktu.

Oproti komplexní knihovně CSP je vývoj minidriveru mnohem snazší, protože je množina implementovaných funkcí (příp. služeb) podstatně méně objemná. Minidriver tvoří dynamicky linkovaná knihovna, která exportuje standardní API pro komunikaci s Microsoft BaseCSP/KSP. Funkce lze klasifikovat do osmi následujících tříd:

Třída bazových funkcí	Množina funkcí
Obecné funkce	<i>CardAcquireContext, CardDeleteContext</i>
Autentizační operace	<i>CardAuthenticatePin, CardGetChallenge, CardAuthenticateChallenge, CardDeauthenticate, CardUnblockPin, CardChangeAuthenticator, CardAuthenticateEx, CardGetChallengeEx, CardDeauthenticateEx, CardChangeAuthenticatorEx</i>
Veřejné datové operace	<i>CardCreateDirectory, CardDeleteDirectory, CardReadFile, CardCreateFile, CardGetFileInfo, CardWriteFile, CardDeleteFile, CardEnumFiles, CardQueryFreeSpace</i>
Schopnosti karty	<i>CardQueryCapabilities</i>
Vlastnosti karty a kontejnerů	<i>CardGetContainerProperty, CardSetContainerProperty, CardGetProperty, CardSetProperty</i>
Správa kontejneru na klíče	<i>CardCreateContainer, CardCreateContainerEx, CardDeleteContainer, CardGetContainerInfo</i>
Kryptografické operace	<i>CardRSADecrypt, CardConstructDHAgreement, CardDeriveKey, CardDestroyDHAgreement, CardSignData, CardQueryKeySizes</i>
Bezpečná injekce klíče	<i>CardGetSharedKeyHandle, CardDestroyKey, CardGetAlgorithmProperty, CardGetKeyProperty, CardSetKeyProperty, CardProcessEncryptedData, CardImportSessionKey</i>

Tab. 10: Funkce modulu minidriver

7.4 Test CNG – KSP

Pro komunikaci s poskytovatelem úložiště kryptografických klíčů lze využít Key Storage API. Funkce jsou účelně zaměřeny na správu a operace s klíči a objekty. Pokrývají všechny podstatné operace, např. de/šifrování, import/export klíče, derivaci klíče, ustanovení sdíleného tajemství a další.

Třída KSP funkcí	Množina funkcí
Obecné funkce	<i>NCryptEnumStorageProviders, NCryptFreeBuffer, NCryptFreeObject, NCryptOpenStorageProvider, NCryptTranslateHandle</i>
Správa klíčů	<i>NCryptCreatePersistedKey, NCryptDeleteKey, NCryptDeriveKey, NCryptEnumAlgorithms, NCryptEnumKeys, NCryptFinalizeKey, NCryptExportKey, NCryptGetProperty, NCryptImportKey, NCryptIsAlgSupported, NCryptIsKeyHandle, NCryptNotifyChangeKey, NCryptOpenKey, NCryptSetProperty</i>
Kryptografické funkce	<i>NCryptDecrypt, NCryptEncrypt, NCryptSecretAgreement, NCryptSignHash, NCryptVerifySignature</i>

Tab. 11: Funkce KSP

8 ROZBOR VÝSLEDKŮ

8.1 Testy PKCS #11

Pomocí testovací aplikace je možné provést řadu integračních testů nad různými implementacemi knihoven PKCS #11. Kryptografické knihovny zároveň spolupracují s konkrétním typem tokenu a některé operace nelze bez přístupových práv vůbec provést. Testy záměrně neuvádějí jména knihoven, jména kryptografických tokenů ani použítá přístupová práva z důvodu ochrany citlivých informací.

8.1.1 Test 1 – Podporované funkce

Třída funkcí	Token #1	Token #2	Token #3	Token #4
Obecné funkce	ANO	ANO	ANO	ANO
Správa slotů a tokenů	ANO	ANO ³	ANO	ANO
Správa sezení	ANO	-	ANO	ANO
Správa objektů	ANO	-	ANO ⁴	ANO ⁷
Správa kryptografických klíčů	ANO ¹	-	ANO ⁵	ANO ¹
Šifrování a dešifrování	ANO	-	ANO	ANO
Kontrolní součty zprávy	ANO	ANO	ANO	ANO
Digitální podpis a MAC	ANO	-	ANO ⁶	ANO
Duální kryptografické funkce	NE	-	NE	ANO
Generování náhodných čísel	ANO ²	ANO	ANO	ANO ²
Paralelní správa funkcí	NE	ANO	ANO	ANO

Tab. 12: Podpora funkcí PKCS #11 tokenů

- 1 – Podpora pouze pro asymetrické klíče RSA.
 2 – Nepodporuje uživatelskou inicializaci generátoru.
 3 – Chyba inicializace uživatelského PIN kódu.
 4 – Chyba při kopírování objektů.
 5 – Nepodporuje *C_UnwrapKey*, nepodporuje algoritmus Diffie-Hellman pro odvození klíče.
 6 – Nepodporuje přímý digitální podpis dat (*C_Sign*).
 7 – Chyba jednotky při vytváření symetrického klíče.

8.1.2 Test 2 – Šifrovací algoritmy

Algoritmus	Token #1	Token #2	Token #3	Token #4
RSA	ANO	ANO	ANO	ANO
AES	NE	ANO	NE	NE
DES	NE	ANO	ANO	ANO
DoubleDES	NE	NE	NE	ANO
TripleDES	NE	ANO	ANO	ANO
RC2	NE	ANO	NE	ANO
RC4	NE	ANO	NE	ANO
DH	NE	ANO	NE	NE

Tab. 13: Podporované šifrovací algoritmy PKCS #11 tokenů

8.1.3 Test 3 – Hashovací algoritmy

Algoritmus	Token #1	Token #2	Token #3	Token #4
MD2	NE	ANO	ANO	ANO
MD5	ANO	ANO	ANO	ANO
SHA-1	ANO	ANO	ANO	ANO
SHA-224	NE	NE	NE	ANO
SHA-256	ANO	ANO	ANO	ANO
SHA-384	NE	NE	ANO	ANO
SHA-512	NE	NE	ANO	ANO
RIPEMD-160	NE	NE	NE	ANO

Tab. 14: Podporované hashovací algoritmy PKCS #11 tokenů

8.1.4 Test 4 – Algoritmy pro digitální podpis/MAC

Algoritmus	Token #1	Token #2	Token #3	Token #4
DSA	NE	ANO	NE	NE
RSA + SHA-1	ANO	ANO	ANO	ANO
RSA + SHA-224	NE	NE	NE	ANO
RSA + SHA-256	ANO	ANO	ANO	ANO
RSA + SHA-384	NE	NE	ANO	ANO
RSA + SHA-512	NE	NE	ANO	ANO
RSA + MD2	NE	ANO	ANO	ANO
RSA + MD5	ANO	ANO	ANO	ANO
RSA + RIPEMD-160	NE	NE	NE	ANO
DES (MAC)	NE	ANO	ANO	ANO
TripleDES (MAC)	NE	ANO	ANO	ANO
RC2 (MAC)	NE	ANO	NE	ANO
RC4 (MAC)	NE	NE	NE	ANO

Tab. 15: Podporované algoritmy PKCS #11 tokenů pro digitální podpis/MAC

8.2 Test CNG – KSP

Pomocí testovací aplikace je možné provést integrační testy poskytovatelů úložiště kryptografických klíčů. Modul KSP může být realizován zcela v software, ale také může aktivně spolupracovat s bezpečným hardware.

8.2.1 Test 1 – Podporované funkce

Třída funkcí	Podmnožina funkcí	KSP #1 (software)	KSP #2 (token 1)	KSP #2 (token 2)
Obecné operace	Obecné funkce	ANO	ANO	ANO
	Seznam poskytovatelů	ANO	ANO	ANO
Správa kryptografických klíčů	Seznam klíčů	ANO	ANO	ANO
	Seznam podporovaných alg.	ANO	ANO	ANO
	Vytváření, mazání	ANO	ANO	ANO
	Export/import	ANO ¹	ANO ²	ANO ²
Kryptografické operace	Šifrování/dešifrování	ANO	ANO	ANO
	Ustanovení sdíleného tajemství	ANO	NE ³	NE ³
	Digitální podpis	ANO	ANO	ANO

Tab. 16: Podpora funkcí KSP

1 – Není podporován export/import privátní části RSA klíče.

2 – Zakázán export/import privátní části RSA klíče.

3 – KSP nepodporuje algoritmus Diffie-Hellman a tokeny nepodporují ECDH algoritmy.

8.2.2 Test 2 – Podporované algoritmy

Algoritmus	KSP #1 (software)	KSP #2 (token 1)	KSP #2 (token 2)
RSA	ANO	ANO	ANO

Tab. 17: KSP - Podporované šifrovací algoritmy

Algoritmus	KSP #1 (software)	KSP #2 (token 1)	KSP #2 (token 2)
DH	ANO	NE	NE
ECDH_P256	ANO	NE ¹	NE ¹
ECDH_P384	ANO	NE ¹	NE ¹
ECDH_P521	ANO	NE ¹	NE ¹

Tab. 18: KSP - Podporované algoritmy pro ustanovení sdíleného tajemství

Algoritmus	KSP #1 (software)	KSP #2 (token 1)	KSP #2 (token 2)
RSA	ANO	ANO	ANO
ECDH_P256	ANO	NE ¹	NE ¹
ECDH_P384	ANO	NE ¹	NE ¹
ECDH_P521	ANO	NE ¹	NE ¹

Tab. 19: KSP - Podporované algoritmy pro digitální podpis

Pozn.: Nenalezeny žádné algoritmy pro symetrické šifrování.

1 – Algoritmy s použitím eliptických křivek nejsou podporovány testovanými tokeny.

ZÁVĚR

Výpočetní výkon počítačů se každým rokem zvyšuje a usnadňuje prolomení starších systémů, které často nepočítaly s určitými riziky do budoucnosti. Při vývoji nových systémů je proto důležité uvážit tato rizika a zvyšovat zabezpečení citlivých informací. Neoprávněnému přístupu zamezíme důsledným nastavením přístupových politik. Také je možné informace utajovat a skrývat. Nejsilnějším dostupným nástrojem je ovšem kryptografie, nauka o šifrování, která nabízí řadu algoritmů pro převod informace do nečitelné podoby. Infrastruktura PKI se pak zabývá zabezpečením na základě asymetrické kryptografie s použitím soukromého a veřejného klíče. Soukromý klíč je nutné důkladně chránit použitím speciálního úložiště nebo HW. Typickým HW představitelem jsou čipové karty, které umožňují relativně bezpečné uložení klíčů ve své interní paměti. Bez znalosti autorizačního kódu PIN není útočníkovi samotná karta příliš platná.

Teoretická část práce rozebírá základy autentizace a identifikace. Čtenář je uveden k základním pilířům kryptografie a dále rozvíjen ve znalostech infrastruktury PKI. Vymezeny jsou klíčové pojmy zaručený elektronický podpis a digitální certifikát. Více technických informací poskytuje kapitola Čipová karta. Předpokladem pro úspěšné zvládnutí praktické části se stala obsáhlá rešerše o programových rozhraních PKI. Mezi nejdůležitější rozhraní patří standardy PKCS #11 a PKCS #15. Na platformě Microsoft Windows se používá systémové rozhraní Cryptography API a novější Cryptography API: Next Generation.

Po nastudování potřebných znalostí z počítačového prostředí byl navržen nástroj pro testování SW knihoven dle standardu PKCS #11. Díky nástroji *PIITest* lze snadno ověřit standardní funkce knihovny s použitím základních kryptografických mechanismů. Důležitým kritériem při vývoji nástroje *PIITest* se ukázala přeložitelnost na více operačních systémech. Druhý testovací nástroj slouží k verifikaci poskytovatelů úložiště kryptografických klíčů (KSP). Program *KSPTest* je již zcela závislý na platformě Microsoft Windows.

Pomocí programových nástrojů proběhla řada testů a jejich výsledky jsou uvedeny v kapitole Rozbor výsledků. Z výsledků jsou patrné výrazné rozdíly mezi různými implementacemi knihoven PKCS #11. Také je patrná závislost na typu čipové karty u KSP s podporou této technologie. Pravidelná exekuce PKI testů přináší dobrý přehled

v konkurenčním prostředí mezi cílovými komerčními subjekty. Na základě získaných výsledků se iniciátor dostává do informačního náskoku o použitých technologiích. Z důvodu ochrany citlivých a komerčních údajů bylo však vhodné uvést pouze anonymní názvy testovaných subjektů.

Ze studia technologií a řešení programů vyplývá fakt, že důležitá systémová rozhraní PKI jsou vzájemně nehomogenní. Existuje několik pokusů o částečné přemostění funkcí z jednoho rozhraní do jiného. Do budoucna lze uvažovat o vývoji univerzálního API, které by pomohlo ke snadnějšímu přenosu technologií. V již tak rozsáhlé struktuře projektů by se potom dobře uplatnila společná vrstva, která by zastřešovala stávající technologie. Cesta k realizaci univerzálního rozhraní však nebude snadná.

CONCLUSION

A computational performance grows every year and this facilitates break through older systems which have not count with some future risks. It is important to consider these risks and try to raise a security of very sensitive information. Consequent settings of an access policy can prevent from an unauthorized access. Also, it is possible to hide information. A science called cryptography is a very powerful tool. The cryptography covers many encryption theories with a lot of algorithms, which converts information to illegible data. A PKI infrastructure deals with the security based on the asymmetric cryptography. It uses private and public keys. The private key has to be protected by a special store or hardware. Smart cards are typical HW representatives. The smart card can safely store keys in their internal memory. This card is not very useful without knowledge of a Personal Identification Code.

Theoretical part analyses basics of authentication and identification. A reader discovers basic pillars of cryptography and he acquires knowledge of the PKI. Also this thesis describes key words like digital signature or digital certificate. More technical information offers a chapter about smart cards. An exploration of facts about PKI interfaces becomes a presumption for a successful practical mastering. The most important standards are PKCS #11 and PKCS #15. Microsoft Windows has its own system interface called Cryptography API and newer Cryptography API: Next Generation.

After necessary studies, an author has developed a tool for testing software libraries in compliance with PKCS #11 standard. Due to *P11Test* tool, it is quite easy to verify intrinsic library functions with essential cryptographic algorithms. Multiplatform program compilation becomes very important criterion for P11Test development. A second test tool can verify key storage providers which are Microsoft Windows platform dependent.

Set of tests sets has been executed due to these tools and corresponding results are evaluated in chapter Rozbor výsledků. Distinctive differences between variant PKCS #11 implementations are obvious. Also a dependency of a smart card type is noticeable for KSP with that support. Regular PKI test execution can bring a great overview of a concurrent environment between target commercial subjects. Based on results, an initiator gets information leading about used technologies. All tested subjects in this thesis are anonymous due to a sensitive information protection.

From PKI technology studies implies that every important PKI interface is heterogeneous. There exists effort for partial by-passing of some functions from one interface to another. We can consider development of universal API which may facilitate technology portability. A common layer would assert in such extensive project structure. It may cover current technologies. United interface realization will not be an easy snap.

SEZNAM POUŽITÉ LITERATURY

1. DOSTÁLEK, Libor; VOHNOUTOVÁ, Marta; KNOTEK, Miroslav. *Velký průvodce infrastrukturou PKI : a technologií elektronického podpisu*. Brno : Computer Press, a.s., 2009. 541 s.
2. Public key infrastructure. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-02]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Public_key_infrastructure>.
3. Kryptografie. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-23]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Kryptografie>>.
4. Symetrická kryptografie. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-23]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Symetrick%C3%A1_kryptografie>.
5. Asymetrická kryptografie. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-23]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Asymetrick%C3%A1_kryptografie>.
6. Diffie-Hellman. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-23]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/Diffie-Hellman>>.
7. RSA. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-23]. Dostupné z WWW: <<http://en.wikipedia.org/wiki/RSA>>.
8. ElGamal. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-23]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/ElGamal>>.
9. Elliptic curve cryptography. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-04-05]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Elliptic_curve_cryptography>.
10. Ministerstvo vnitra České republiky. *Ministerstvo vnitra České republiky* [online]. 2010 [cit. 2011-03-24]. Zákon č. 227/2000 Sb., o elektronickém podpisu. Dostupné z WWW: <<http://www.mvcr.cz/clanek/zakon-c-227-2000-sb-o-elektronickem-podpisu.aspx>>.
11. Elektronický podpis. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-24]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Elektronick%C3%BD_podpis>.
12. Digitální certifikát. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Digit%C3%A1ln%C3%AD_certifik%C3%A1t>.

13. *ITU-T X.509*. Switzerland, Geneve : International telecommunication union, 2005. 174 s.
Dostupné z WWW: <<http://www.itu.int/rec/T-REC-X.509>>.
14. Smart card. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-28]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Smart_card>.
15. ISO/IEC 7816-3:2006, Identification cards - Integrated circuit cards - Part 3: Cards with contacts – Electrical interface and transmission protocols
16. Smart card application protocol data unit. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-28]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Smart_card_application_protocol_data_unit>.
17. Pretty Good Privacy#OpenPGP. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-29]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Pretty_Good_Privacy#OpenPGP>.
18. PKCS. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-29]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/PKCS>>.
19. PKCS #11 v2.20: Cryptographic Token Interface Standard, RSA Laboratories, 28 June 2004, dostupné z <<ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf>>
20. PKCS #15 v1.1: Cryptographic Token Information Syntax Standard, RSA Laboratories, June 6, 2000, dostupné z ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-15/pkcs-15v1_1.pdf
21. Abstract Syntax Notation One. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-31]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One>.
22. Distinguished Encoding Rules. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-03-31]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Distinguished_Encoding_Rules>.
23. Microsoft. *MSDN* [online]. 2011 [cit. 2011-04-04]. Cryptography Essentials. Dostupné z WWW: <<http://msdn.microsoft.com/en-us/library/aa380251%28v=VS.85%29.aspx>>.
24. Microsoft. *MSDN* [online]. 2011 [cit. 2011-04-04]. Cryptographic Service Providers. Dostupné z WWW: <<http://msdn.microsoft.com/en-us/library/aa380245%28v=VS.85%29.aspx>>.
25. Microsoft. *MSDN* [online]. 2011 [cit. 2011-04-04]. Cryptography API: Next Generation. Dostupné z WWW: <<http://msdn.microsoft.com/en-us/library/aa376210%28v=VS.85%29.aspx>>.
26. NSA Suite B Cryptography. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-04-05]. Dostupné z WWW: <http://en.wikipedia.org/wiki/NSA_Suite_B_Cryptography>.

27. Microsoft Corporation. *Windows Smart Card Minidriver Specification* [online]. Version 7.06. [s.l.] : [s.n.], 2009 [cit. 2011-04-04]. Dostupné z WWW: <<http://www.microsoft.com/whdc/device/input/smartcard/sc-minidriver.msp>>.
28. Apple Inc. *Security Architecture* [online]. 2010 [cit. 2011-05-05]. Dostupné z WWW: <http://developer.apple.com/library/ios/#documentation/Security/Conceptual/Security_Overview/Architecture/Architecture.html>.
29. Apple Inc. *SmartCard Services : TokenD* [online]. 2011 [cit. 2011-05-07]. Dostupné z WWW: <<http://smartcardservices.macosforge.org/trac/wiki/tokend>>.
30. Java Card. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2011-05-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Java_Card>.
31. *Java Card Platform Specification*. 3.0.1. Santa Clara, California, U.S.A. : Sun Microsystems, Inc., 2009, [cit. 2011-05-08]. Dostupné z WWW: <<http://www.oracle.com/technetwork/java/javacard/documentation/index.html>>.
32. *OpenSC* [online]. 2011 [cit. 2011-04-20]. Dostupné z WWW: <<http://www.opensc-project.org/opensc>>.
33. *Mozilla : Open Source PKI Projects* [online]. 2011 [cit. 2011-05-08]. Dostupné z WWW: <<http://www.mozilla.org/projects/security/pki/>>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
PKI	Public Key Infrastructure
ICC	Integrated Circuit Card
PC/SC	Personal Computer/Smart Card
PKCS	Public Key Cryptographic Standards
CSP	Cryptographic Service Provider
CNG	Cryptography API: Next generation
KSP	Key Store Provider
SSL	Secure Sockets Layer
RSA	Rivest-Shamir-Adleman
AES	Advanced Encryption Standard
DSA	Digital Signature Algorithm
SHA	Secure Hash Algorithm
MD	Message Digest
ECC	Elliptic-Curve Cryptography
ECDH	Elliptic-Curve Diffie-Hellman
ECDSA	Elliptic-Curve Digital Signature Algorithm
CA	Certificate authority, certifikační autorita
SSO	Single sign-on
PMI	Privilege Management Infrastructure
CRL	Certificate Revocation List
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CER	Canonical Encoding Rules
DER	Distinguished Encoding Rules
BCD	Byte Coded Decimal
APDU	Application Protocol Data Unit
PIN	Personal Identification Number
PUK	PIN Unlock Key
JCE	Java Cryptography Extension
SO	Security Officer
MAC	Message Authentication Code (v sítích také Media Access Control)
CryptoAPI	Cryptography Application Programming Interface
CryptoSPI	Cryptography System Program Interface
CNG	Cryptography API: Next generation
SDK	Software Development Kit
JCVM	Java Card Virtual Machine
JCRE	Java Card Runtime Environment
NSS	Network Security Services (Mozilla)
JSS	Network Security Services for Java (Mozilla)
PSM	Personal Security Manager (Mozilla)

SEZNAM OBRÁZKŮ

Obr. 1: Princip symetrického šifrování	14
Obr. 2: Princip asymetrického šifrování	15
Obr. 3: Vydání a použití digitálního certifikátu	18
Obr. 4: Princip digitálního podpisu [11]	22
http://cs.wikipedia.org/wiki/Soubor:Digital_Signature_diagram_cs.svg	
Obr. 5: ID-1 čipová karta [14]	26
http://commons.wikimedia.org/wiki/File:Smartcard2.png	
Obr. 6: Vývody kontaktní čipové karty [14]	27
http://en.wikipedia.org/wiki/File:SmartCardPinout.svg	
Obr. 7: USB čtečka čipových karet	28
http://fingerprint-security.net/wp-content/uploads/2010/12/smart-card-reader.jpg	
Obr. 8: Obecný Cryptoki model [19]	32
Obr. 9: Cryptoki - stavy R/O sezení [19]	33
Obr. 10: Cryptoki - stavy R/W sezení [19]	33
Obr. 11: Hierarchie PKCS #15 objektů [20]	34
Obr. 12: Typická topologie reprezentace objektů PKCS #15 [20]	35
Obr. 13: Architektura CryptoAPI – CSP [24]	36
http://msdn.microsoft.com/en-us/library/aa380239%28v=VS.85%29.aspx	
Obr. 14: Architektura CSP [24]	37
http://msdn.microsoft.com/en-us/library/aa381482%28v=VS.85%29.aspx	
Obr. 15: CNG - Architektura KSP [25]	39
http://msdn.microsoft.com/en-us/library/bb204778%28v=VS.85%29.aspx	
Obr. 16: Topologie CNG a minidriverů [27]	41
http://msdn.microsoft.com/en-us/windows/hardware/gg487500.aspx	
Obr. 17: Minidriver - Bezpečná injekce klíče [27]	43
http://msdn.microsoft.com/en-us/windows/hardware/gg487500.aspx	
Obr. 18: Architektura CDSA a CSSM [28]	44
http://developer.apple.com/library/ios/#documentation/Security/Conceptual/Security_Overview/Architecture/Architecture.html	
Obr. 19: Přehled projektu OpenSC [32]	47
http://www.opensc-project.org/opensc/attachment/wiki/OverView/OpenSC.png	

SEZNAM TABULEK

Tab. 1: Vývody kontaktní čipové karty.....	27
Tab. 2 Typy APDU protokolů [15].....	29
Tab. 3 APDU příkaz [16].....	29
Tab. 4 APDU odpověď [16]	30
Tab. 5: Cryptoki - stavy R/O sezení [19].....	33
Tab. 6: Cryptoki - stavy R/W sezení [19]	34
Tab. 7: Funkce PKCS #11	52
Tab. 8: Roztřídění funkcí CryptoAPI.....	53
Tab. 9: Funkce CryptoAPI - základní kryptografie.....	54
Tab. 10: Funkce modulu minidriver	55
Tab. 11: Funkce KSP	55
Tab. 12: Podpora funkcí PKCS #11 tokenů.....	56
Tab. 13: Podporované šifrovací algoritmy PKCS #11 tokenů.....	56
Tab. 14: Podporované hashovací algoritmy PKCS #11 tokenů.....	57
Tab. 15: Podporované algoritmy PKCS #11 tokenů pro digitální podpis/MAC	57
Tab. 16: Podpora funkcí KSP	58
Tab. 17: KSP - Podporované šifrovací algoritmy.....	58
Tab. 18: KSP - Podporované algoritmy pro ustanovení sdíleného tajemství	58
Tab. 19: KSP - Podporované algoritmy pro digitální podpis.....	58

SEZNAM PŘÍLOH

PŘÍLOHA 1: CD-ROM

PŘÍLOHA 1: CD-ROM

Přiložený CD-ROM obsahuje:

1. spustitelný testovací nástroj *PIITest*,
2. spustitelný testovací nástroj *KSPTest*,
3. zdrojové texty pro překlad testovacích nástrojů
(vypracováno v rámci spolupráce autora, Univerzity Tomáše Bati ve Zlíně
a firmy Monet+, a.s.),
4. elektronický dokument obsahující text diplomové práce.