

Žilinská univerzita v Žiline

Fakulta riadenia a informatiky

DIPLOMOVÁ PRÁCA

Študijný odbor: Informačné systémy
Študijný program: Aplikovaná informatika

Bc. Matej Kurpel

**Kryptografické zariadenie
vo forme mobilného telefónu
Mobile phone acting as a cryptographic token**

Vedúci: doc. Ing. Emil Kršák, PhD.

Reg. č. 180/2010 reg. dňa 25. 10. 2010

Žilina

ŽILINSKÁ UNIVERZITA V ŽILINE, FAKULTA RIADENIA A INFORMATIKY

ZADANIE TÉMY DIPLOMOVEJ PRÁCE

Študijný odbor: Informačné systémy

Zameranie: Aplikovaná informatika

Meno a priezvisko

Matej Kurpel

Osobné číslo

552078

Názov práce v slovenskom aj anglickom jazyku

**Kryptografické zariadenie vo forme mobilného telefónu
Mobile phone acting as a cryptographic token**

Zadanie úlohy, ciele, pokyny pre vypracovanie

Cieľ diplomovej práce:

Cieľ diplomovej práce je navrhnuť a implementovať programové vybavenie pre PC a mobilný telefón s platformou Windows Mobile, poskytujúce služby pre digitálny podpis a šifrovanie. Realizácia bude spĺňať štandard PKCS#11. Riešenie bude umožňovať využitie vo všetkých aplikáciách na PC, ktoré využívajú štandard PKCS#11.

Obsah:

Riešenie by malo obsahovať nasledujúce vlastnosti a funkcionality:

- PKCS#11 modul pre PC aplikácie schopné používať toto rozhranie
- Návrh komunikácie medzi PKCS#11 modulom a softvérom v telefóne
- Certifikáty a súkromné kľúče uložené v telefóne
- Správa párov certifikát + súkromný kľúč v softvéri na strane telefónu
- Symetrické šifrovanie súkromného kľúča pre vyššiu bezpečnosť
- Digitálny podpis a šifrovanie testovať na e-mailovom klientovi Mozilla Thunderbird

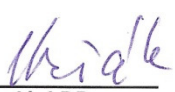
Témy z predmetov študijného zamerania

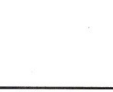
5SI30: 1, 2, 3

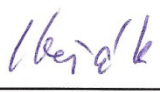
Meno a pracovisko vedúceho DP:


doc.Ing. Emil Kršák PhD., FRI - KST, ŽU

Meno a pracovisko tútora DP:

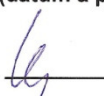

vedúci DP
(dátum a podpis)


tútor
(dátum a podpis)


vedúci katedry
(dátum a podpis)


garant
(dátum a podpis)

Zadanie zaregistrované dňa 25. 10. 2010 pod číslom 180/2010 podpis



POĎAKOVANIE

Ďakujem vedúcemu svojej práce doc. Ing Emilovi Kršákovi, PhD. za jeho odbornú pomoc, cenné rady, metodické vedenie a za zapožičanie prístroja HP iPAQ h2200 na účely testovania vytvoreného softvéru. Vďaka patrí aj mojej rodine za podporu počas celého štúdia na Fakulte riadenia a informatiky Žilinskej univerzity v Žiline.

ABSTRAKT

KURPEL, Matej: *Kryptografické zariadenie vo forme mobilného telefónu* [diplomová práca] – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra softvérových technológií. – Vedúci: doc. Ing. Emil Kršák, PhD. – Stupeň odbornej kvalifikácie: Inžinier v študijnom programe Aplikovaná informatika. Žilina: FRI ŽU v Žiline, 2011. – 70 s.

Cieľom diplomovej práce bolo navrhnuť a implementovať také softvérové vybavenie, ktoré by umožňovalo použitie mobilného telefónu ako bežného kryptografického tokenu.. V teoretickej časti práce sú vysvetlené základné použité pojmy z oblasti kryptografie, so zameraním na asymetrickú kryptografiu. Popísané sú procesy šifrovania, digitálneho podpisu a systém dôvery v PKI. V ďalšej časti je rozobratá analýza a návrh riešenia s popisom rozdelenia softvéru na dve komunikujúce strany. Nasleduje popis fungovania oboch strán. Nakoniec je rozobratá metodika testovania tohto softvéru počas vývoja a stručný popis inštalácie a nastavenia z používateľského pohľadu (overenie riešenia).

Kľúčové slová: kryptografia, kľúč, certifikát, mobilný telefón, šifrovanie, digitálny podpis

ABSTRACT

KURPEL, Matej: *Mobile phone acting as a cryptographic token* [diploma thesis] – The University of Žilina. Faculty of Management Science and Informatics; Department of Software Technologies. – Tutor: doc. Ing. Emil Kršák, PhD. – Qualification level: Engineer in study program Applied Informatics. Žilina: FRI ŽU in Žilina, 2011. – 70 p.

The aim of this diploma thesis was to design and implement software equipment to enable using a mobile phone as a regular cryptographic token. In the theoretical part, basic terms from cryptography are explained, with the focus on asymmetric cryptography. The processes of encryption, producing a digital signature and distribution of trust in PKI are described. In the next part, analysis and design of problem solution is explained, with the details of both communicating parts of the software, and the description of how both parts work. At last, the methods of testing during development are covered, and a brief guide of installing the software from user's standpoint is provided.

Keywords: cryptography, key, certificate, mobile phone, encryption, digital signature

PREDHOVOR

Bezpečnosť komunikácie je ešte stále oblasť, ktorá sa dostáva do povedomia verejnosti len veľmi pozvoľna. Drvivá väčšina služieb vyžaduje na prihlásenie len užívateľské meno a heslo, ktoré si ľudia zvyknú voliť jednoduché a prípadne ešte pre viacero služieb rovnaké. E-maily sa posielajú nepodpísané a nešifrované, čo vystavuje celú komunikáciu bezpečnostnému riziku.

Alternatívou k používaniu užívateľského mena a hesla je použitie súkromného kľúča danej osoby a prislúchajúceho osobného certifikátu. Toto je vec, ktorá sa žiaľ veľmi nevyužíva, pretože kvôli maximalizácii spoľahlivosti takéhoto systému autentifikácie je nutné zvláštne hardvérové vybavenie - čítačka bezpečnostných tokenov a samotný token, čo nie je práve lacná záležitosť. Navyiac, inštalácia a nastavenie systému i tokenu zvyčajne nie je dostatočne jednoduché na to, aby to bez problémov zvládol bežný užívateľ.

Spomínané bezpečnostné zariadenia zvládajú okrem autentifikácie aj mnoho ďalších činností, ako je napríklad digitálne podpisovanie dokumentov či e-mailov, šifrovanie, generovanie kľúčov a ďalšie.

Dnes už snáď každý vlastní mobilný telefón. Myšlienka, že práve mobilný telefón by mohol slúžiť ako kryptografický token, sa mi zapáčila a preto som si určil, že sa v rámci svojej diplomovej práce budem venovať jej rozvíjaniu a skúmaniu do takej podoby, aby výsledkom bol softvér pripravený na použitie. Keďže sa jedná o dosiaľ veľmi málo preskúmanú oblasť, bolo pre mňa výzvou v nej bádať.

Tvorbou softvéru v rámci tejto diplomovej práce som sa naučil mnoho nových vecí, najmä z praktickej časti kryptografie (reprezentácia kľúčov a certifikátov v počítači atď.), a do značnej miery rozvinul svoje poznatky ohľadom programovania v jazykoch C++ a C#.NET.

Čestne vyhlasujem, že som diplomovú prácu vypracoval samostatne s využitím vlastných teoretických poznatkov i praktických skúseností, získaných nielen samoštúdiom, ale i počas štúdia na Fakulte riadenia a informatiky.

OBSAH

1	ÚVOD.....	1
2	TEORETICKÝ ÚVOD DO PROBLEMATIKY	2
2.1	ZÁKLADNÉ POJMY A PRINCÍPY	2
2.1.1	Kryptológia a pridružené vedné odbory	2
2.1.2	Šifrovanie - princíp	2
2.1.3	Digitálny podpis - princíp	4
2.1.4	PKI	5
2.2	AKO TO FUNGUJE V PRAXI	7
2.2.1	Šifrovanie v praxi	7
2.2.2	Digitálny podpis v praxi	9
2.3	KRYPTOGRAFICKÉ ZARIADENIA, TOKENY	10
2.3.1	Rozhrania prístupu k tokenom	11
2.4	KRYPTOGRAFICKÉ ŠTANDARDY	12
2.4.1	Súbor štandardov PKCS	12
2.4.1.1	PKCS#11	12
2.5	ALGORITMY	15
2.5.1	Kryptosystém RSA	15
2.5.1.1	RSA - generovanie kľúčov	15
2.5.1.2	RSA - šifrovanie	15
2.5.1.3	RSA - dešifrovanie	15
2.5.2	Hashovacie algoritmy	16
2.5.3	Kódovanie Base64	17
2.5.4	ASN.1 a DER kódovanie	18
2.6	FORMÁTY	20
2.6.1	Formáty certifikátov	20
2.6.2	Formáty súkromných kľúčov	21
2.6.3	Formáty verejných kľúčov	23
3	ANALÝZA A NÁVRH RIEŠENIA	24
3.1	CELKOVÝ POHĽAD NA RIEŠENIE.....	24
3.2	NÁVRH KOMUNIKÁCIE	27

3.3	POUŽITÉ PRODUKTY TRETÍCH STRÁN	30
3.3.1	Použitý kód tretích strán na strane PC	30
3.3.1.1	TinyXML	30
3.3.1.2	Base64 kódovač a dekodovač	30
3.3.1.3	MD5	30
3.3.2	Použitý kód tretích strán na strane mobilného telefónu	31
3.3.2.1	Funkcie pre vyvolanie natívnych CryptoAPI funkcií	31
3.3.2.2	Detekcia výrobcu a platformy mobilného zariadenia	31
3.3.2.3	Metódy na prácu s kľúčmi	31
4	SOFTVÉR NA STRANE TELEFÓNU	32
4.1	POPIS TRIED A METÓD	33
4.1.1	Projekt CertKeyLib	34
4.1.2	Projekt XMLBuildLib	35
4.1.3	Projekt CrySign Mobile	35
4.1.4	Projekt PKCS11Lib	37
4.1.5	Projekt PInvokeLib	38
4.2	SPRÁVA CERTIFIKÁTOV A KĹÚČOV	39
5	SOFTVÉR NA STRANE PC	41
5.1	VŠEOBECNÝ POHĽAD NA PKCS#11 FUNKCIE	41
5.2	NÁJDENIE MOBILNÉHO ZARIADENIA	43
5.3	KONFIGURAČNÝ SÚBOR PKCS#11 MODULU	44
6	TESTOVANIE POČAS VÝVOJA	46
6.1	TESTOVANIE NA STRANE PC	46
6.2	TESTOVANIE NA STRANE MOBILNÉHO ZARIADENIA	48
7	PODROBNOSTI VOLANIA PKCS#11 FUNKCIÍ	49
7.1	SPUSTENIE PKCS#11 PROGRAMU	49
7.2	ZOBRAZENIE INFORMÁCIÍ O TOKENE A SLOTE	50
7.3	ZOBRAZENIE ZOZNAMU CERTIFIKÁTOV	51
7.4	ODOSLANIE DIGITÁLNE PODPÍSANÉHO E-MAILU	51
7.5	ČÍTANIE ZAŠIFROVANÉHO E-MAILU	52
7.6	PRIHLÁSENIE POMOCOU OSOBNÉHO CERTIFIKÁTU NA WEBE	52
7.7	GENEROVANIE RSA KĹÚČOV	53
8	OVERENIE RIEŠENIA	55
8.1	INŠTALÁCIA A NASTAVENIE SOFTVÉRU NA STRANE TELEFÓNU	55

8.2	NASTAVENIE E-MAIL. KLIENTA MOZILLA THUNDERBIRD	57
8.2.1	Inštalácia PKCS#11 modulu	57
8.2.2	Inštalácia certifikátu certifikačnej autority	58
8.2.3	Predvolenie osobného certifikátu pre šifrovanie a digitálny podpis	59
8.3	ODOSLANIE A PRIJATIE DIGITÁLNE PODPÍSANÉHO E-MAILU	61
8.4	ODOSLANIE A PRIJATIE ZAŠIFROVANÉHO E-MAILU	63
8.5	PRIHLÁSENIE POMOCOU OSOBNÉHO CERTIFIKÁTU NA WEBE	64
8.6	GENEROVANIE RSA KĹÚČOV	65
9	ZÁVER	67
10	ZOZNAMY	68
10.1	ZOZNAM POUŽITÝCH SKRATIEK	68
10.2	ZOZNAM POUŽITÝCH ZDROJOV	69
10.3	ZOZNAM PRÍLOH	70
11	PRÍLOHY	71
	Príloha č. 1	70
	Príloha č. 2	74
	Príloha č. 3	75
	Príloha č. 4	76
	Príloha č. 5	77
	Príloha č. 6	81
	Príloha č. 7	83
	Príloha č. 8	87
	Príloha č. 9	91
	Príloha č. 10	95

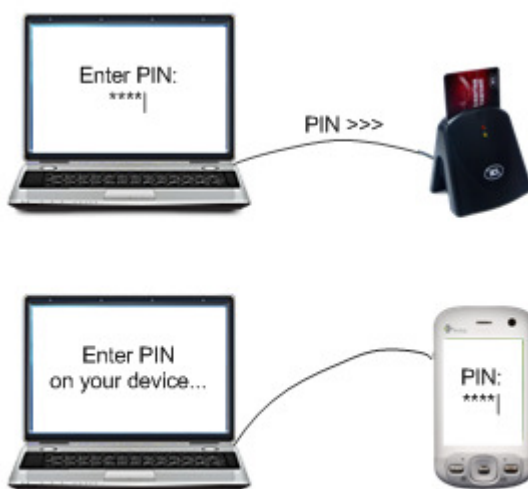
1 ÚVOD

Moderné počítačové programy využívajúce služby kryptografických tokenov zväčša ponúkajú aj možnosť uložiť osobný certifikát a súkromný kľúč v počítači. Toto odbúrava nutnosť použitia špeciálneho hardvéru, ale na druhej strane otvára dvere útočníkom. Programy majú bezpečnostné chyby a tie môže útočník zneužiť, aby sa dostal k súkromnému kľúču nič netušiaceho užívateľa. Prípadne si ešte pomocou programu na odchyťovanie stlačených kláves zistí heslo k tomuto súkromnému kľúču, čo mu umožní vystupovať pod identitou obete.

Preto základný princíp použitia hardvérových tokenov je jasný: oddeliť bezpečné úložisko od potenciálne nebezpečného prostredia. Je samozrejmé, že citlivé informácie, ako sú súkromné kľúče, nikdy neopustia prostredie tohto tokenu. Token má bežne podobu čipovej karty.

Súkromné kľúče bývajú uložené na tokene v zašifrovanej podobe, preto aj keď ho jeho držiteľ stratí alebo on inak príde, jeho súkromné údaje sú v bezpečí pokiaľ neprezradil svoje heslá k príslušným súkromným kľúčom. Súkromný kľúč sa dešifruje len pred jeho použitím, a na tento účel si čítačka tokenu vypýta od užívateľa heslo. Existujú čítačky, ktoré nemajú klávesnicu ani inú metódu vstupu od užívateľa, a tak si pýtajú heslo cez počítač - heslo je vystavené riziku odchytenia nežiaducimi osobami.

Každý mobilný telefón má klávesnicu (hardvérovú či softvérovú), ktorá sa dá s prehľadom využiť na vstup hesla k súkromnému kľúču. Názornejšie porovnanie tokenu s čítačkou bez klávesnice a mobilného telefónu je zobrazené na obrázku 1-1.



Obr. 1-1: Porovnanie čítačky tokenu bez klávesnice s mobilným telefónom

2 TEORETICKÝ ÚVOD DO PROBLEMATIKY

Táto kapitola poskytuje všeobecný úvod do problematiky kryptografie a jej praktického využitia. Popisuje fungovanie šifrovania a digitálneho podpisu i spôsob rozloženia dôvery v PKI. Kapitola 2.3 je venovaná kryptografickým zariadeniam a tokenom, a tiež spôsobom prístupu k nim z aplikácií bežiacich na počítači.

2.1 ZÁKLADNÉ POJMY A PRINCÍPY

2.1.1 Kryptológia a pridružené vedné odbory

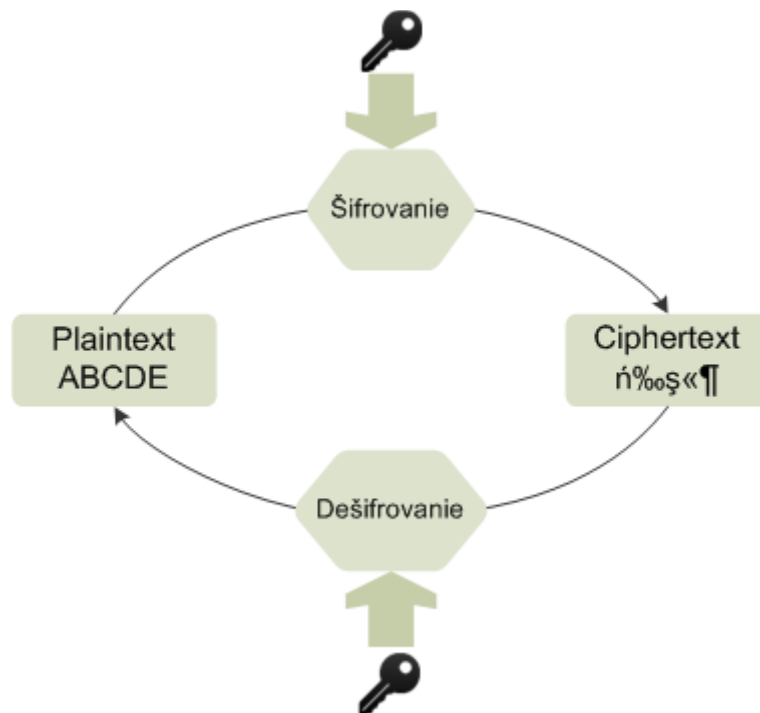
Podľa [1], kryptológia (z gréčtiny zo slov κρυπτός [kryptós], čo znamená ukryť a γράφω [gráfo], čo znamená písať alebo λεγειν [legein] „hovoríť“) je veda o utajení obsahov správ. Je používaná v technologicky vyspelých aplikáciách; napríklad v zabezpečení peňažných bankomatov, počítačových hesiel a v elektronickom obchode. Kryptológia sa rozdeľuje na kryptografiu, kryptoanalýzu a steganografiu [1].

- *Kryptografia* je oblasť kryptológie, ktorej cieľom je skúmanie a navrhovanie šifrovacích systémov, ktoré budú spĺňať určité podmienky ako autenticitu, dôvernosť, dostupnosť, integritu a pôvod dát. Ich úlohou je teda urobiť určitý obsah správy nečitateľným v prípade jeho zachytenia treťou osobou.
- *Kryptoanalýza* je oblasť kryptológie, ktorej cieľom je skúmanie metód lúštenia šifrovacích systémov (kryptosystémov), čiže je to rozbor zašifrovaných správ a ich metód šifrovania, ktorými sa hľadá spôsob preniknutia do tohto kryptosystému.
- *Steganografia* je oblasť kryptológie, ktorej cieľom je zatajiť existenciu danej správy (do inej správy). Steganografické metódy obsahujú napríklad ukrytie správy do obrázka a pod.

2.1.2 Šifrovanie - princíp

Šifrovanie je utajenie informácie použitím kľúča. Spôsob šifrovania a dešifrovania je daný kryptosystémom. Aby sme mohli utajenú informáciu dešifrovať, musíme poznať kľúč. Čo môže poslúžiť ako kľúč, je opäť závislé od použitého kryptosystému. Môže to byť číslo, reťazec, postupnosť 8-bitových čísel a pod.

Správa vstupujúca do procesu šifrovania sa nazýva *priamy text* (alebo *plaintext*). Výsledok - zašifrovaná správa sa nazýva *zašifrovaný text* (*ciphertext*).



Obr. 2-1: Vzťah medzi priamym a zašifrovaným textom

Poznáme šifrovanie *symetrické* a *asymetrické*. Pri symetrickom šifrovaní sa na šifrovanie aj dešifrovanie používa ten istý kľúč. Asymetrické šifrovanie používa dvojicu kľúčov, pričom platí, že čo zašifrujeme jedným kľúčom, je možné dešifrovať len druhým kľúčom z tejto dvojice. V praxi jeden z kľúčov zverejníme (*verejný kľúč*), druhý utajíme (*súkromný kľúč*).

Kryptosystém, ktorý sa používa, určuje:

- asymetrickosť či symetrickosť šifrovania a dešifrovania,
- množinu povolených kľúčov,
- spôsob, akým sa z priameho textu stane zašifrovaný text a naopak,
- či sa priamy text šifruje znak po znaku, bit po bite, alebo po blokoch.

Podľa Kerckhoffovho princípu [2], dobrý kryptosystém sa vyznačuje tým, že prezradenie šifrovacieho algoritmu nemá vplyv na bezpečnosť systému. To znamená, že bezpečnosť spočíva iba v utajení kľúča.

Laická verejnosť si často mýli pojmy *šifrovanie* a *kódovanie*, preto ich treba na tomto mieste ujasniť.

- *Šifrovanie* je utajenie informácie.
- *Kódovanie* je úprava informácie do tvaru potrebného pre ďalšie spracovanie (odoslanie prenosovým kanálom, uloženie niekam a pod.).

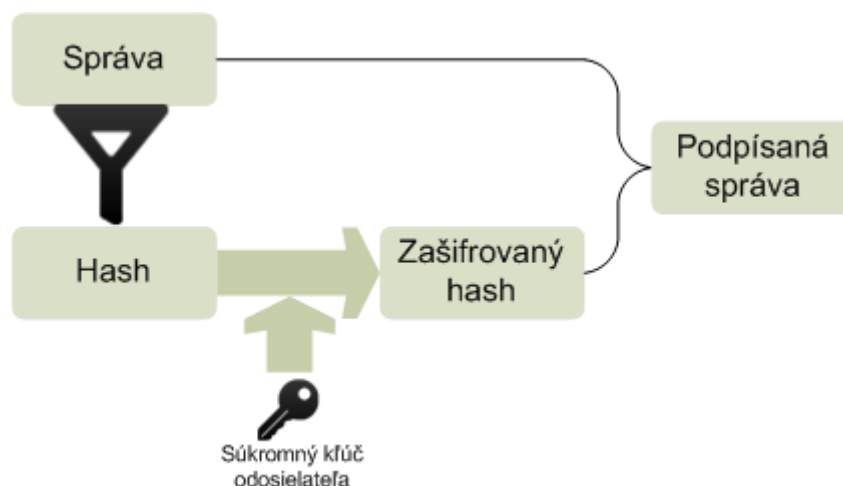
2.1.3 Digitálny podpis - princíp

Kryptografické metódy nachádzajú veľmi časté uplatnenie pri digitálnom podpisovaní správ. Digitálny podpis sú dáta priložené k správe, ktoré zaručujú [3]:

- *autenticitu*: možnosť overenia identity osoby, ktorá správu podpísala,
- *integritu*: správa nebola poškodená či upravená po jej podpísaní,
- *nepopierateľnosť*: osoba, ktorá správu podpísala, nemôže poprieť, že tak urobila,
- *možnosť overenia času podpisu*: ak správa obsahuje aj časovú pečiatku.

Na digitálne podpisovanie správ sa používajú asymetrické kryptosystémy. Odosielateľ použije na podpísanie správy svoj súkromný kľúč, čo zaisťuje, že on je jediný, kto mohol danú správu podpísať.

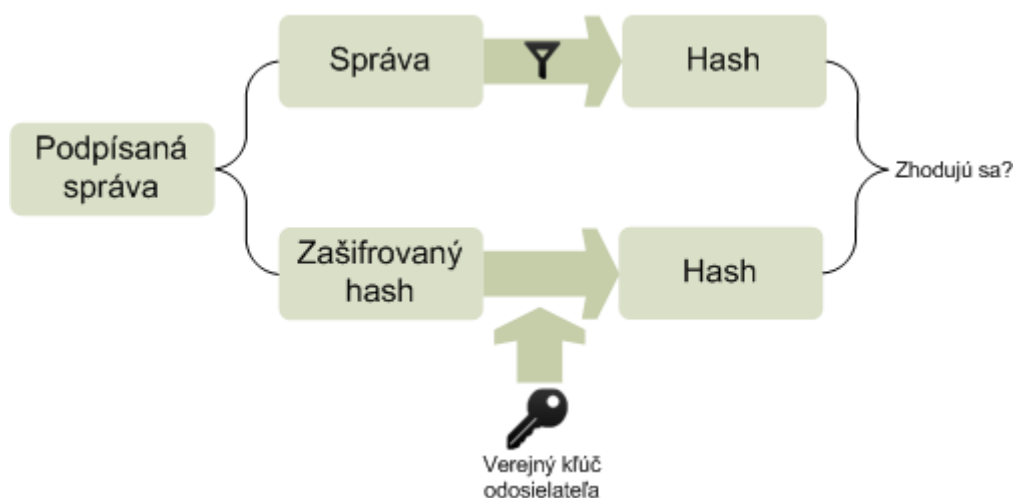
Podpísanie správy funguje tak, že najprv sa vytvorí zo správy jej odtlačok - tzv. *hash* (viac v kapitole 2.5.2) - ten je pre každú správu iný. Tento hash sa zašifruje súkromným kľúčom odosielateľa a pripojí sa k správe. Obe časti spolu tvoria podpísanú správu, ktorá sa odošle adresátovi.



Obr. 2-2: Digitálny podpis na strane odosielateľa správy

Príjemca správy oddelí zašifrovaný hash od tela správy. Tento hash dešifruje verejným kľúčom odosielateľa¹. Zároveň vypočíta hash správy samotnej. Ak sa tento hash zhoduje s hashom, ktorý bol dešifrovaný verejným kľúčom odosielateľa, potom je digitálny podpis platný. Tento proces je znázornený na obrázku 2-3.

¹ Verejný kľúč odosielateľa sa prikladá k podpísanej správe, čo sa tu pre jednoduchosť neuvádza. Spôsob odoslania podpísanej správy a overenia podpisu v praxi môže čitateľ nájsť v kapitole 2.2.2.



Obr. 2-3: Overenie digitálneho podpisu u príjemcu správy

2.1.4 PKI

PKI (Public Key Infrastructure) je označenie infraštruktúry správy a distribúcie verejných kľúčov. Umožňuje používať cudzie verejné kľúče a overovať ich pravosť bez nutnosti individuálnej kontroly [4].

V doteraz popísaných schémach nebola možnosť overiť identitu vlastníka verejného kľúča. Jedným zo spôsobov, ako to zariadiť, je prostredníctvom certifikačných autorít (CA). To znamená, že certifikačná autorita sa zaručí, že daný verejný kľúč patrí práve tej konkrétnej osobe, zväčša osobnou identifikáciou pomocou dokladov ako je občiansky preukaz. Certifikačná autorita vydá *digitálny certifikát*, čo je súbor obsahujúci verejný kľúč danej osoby, podpísaný súkromným kľúčom certifikačnej autority. Certifikát zvykne obsahovať ešte ďalšie položky:

- dátum platnosti,
- sériové číslo,
- účely, na ktoré môže byť daný verejný kľúč použitý,
- webovú adresu certifikačnej autority a CRL² či OCSP³

Ak dôverujeme danej certifikačnej autorite, potom dôverujeme všetkým certifikátom, ktoré vydala a ktoré sú v platnosti. Tieto certifikáty dokazujú, že verejné

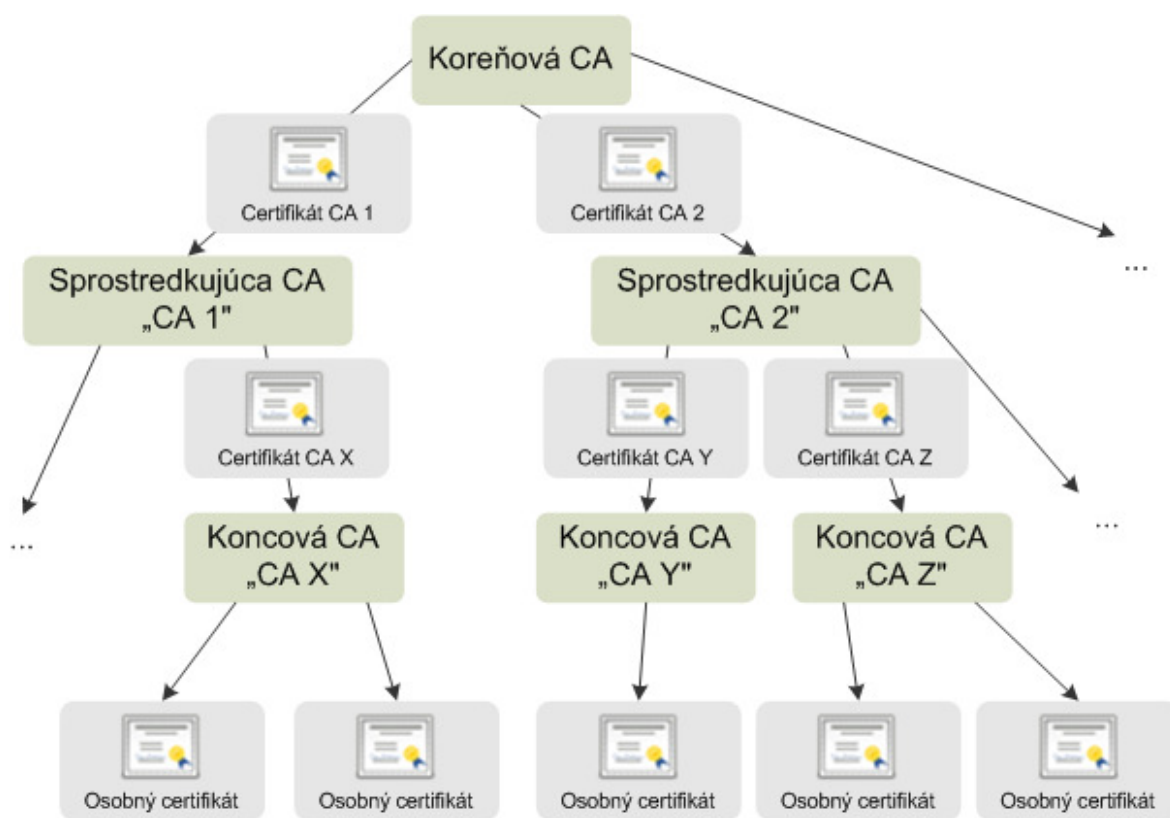
² Certificate Revocation List - zoznam sériových čísel odvolaných certifikátov. Vydáva ho certifikačná autorita a uvádza certifikáty zneplatnené pred uplynutím doby platnosti, napríklad kvôli úniku kľúčov daného subjektu.

³ Online Certificate Status Protocol - internetový protokol slúžiaci na dopytovanie sa certifikačnej autority na platnosť certifikátu s konkrétnym sériovým číslom.

klúče v nich obsiahnuté naozaj patria uvedeným osobám - za to sa zaručila certifikačná autorita tým, že tieto certifikáty podpísala.

Bolo by možné položiť si otázku, kto zodpovedá za to, že sama certifikačná autorita je tým, za koho sa vydáva? Odpoveďou je, že za to zodpovedá v hierarchii certifikačných autorít jej nadradená certifikačná autorita. CA, ktorá je na vrchole hierarchie, sa považuje za všeobecne dôveryhodnú a keďže už nie je nad ňou žiadna CA, ktorá by podpísala jej vlastný certifikát, podpíše si ho sama.

Webové prehliadače, či iné programy, i operačný systém samotný, majú koreňové certifikáty najväčších certifikačných autorít už predinštalované. Na to, aby sme dôverovali koncovému certifikátu, stačí dôverovať certifikačnej autorite, ktorá ho vydala. Ak to nie je možné, nasleduje kontrola dôvery u CA, ktorá vydala jej certifikát a tak ďalej až k vrcholu hierarchie (každý certifikát obsahuje polia „vydaný kým“ a „vydaný komu“). Dôverovať certifikačnej autorite znamená nainštalovať si jej vlastný certifikát do zoznamu dôveryhodných certifikačných autorít v danom softvéri, či priamo do úložiska v operačnom systéme na to určeného; závisí od toho, ktoré z týchto úložísk používa nami využívaný softvér.



Obr. 2-4: Znáoznenie hierarchie certifikačných autorít

2.2 AKO TO FUNGUJE V PRAXI

Táto kapitola sa na rozdiel od kapitoly 2.1 zaoberá aplikáciou spomenutých princípov v praxi - konkrétne pri digitálnom podpise a šifrovaní e-mailov. Oba procesy sú popísané podľa toho, ako ich vykonáva e-mailový klient Mozilla Thunderbird (používajúci štandard S/MIME [5]). Čitateľ potrebuje mať znalosti o fungovaní PKI (časť 2.1.4).

2.2.1 Šifrovanie v praxi

Keď chceme niekomu poslať šifrovaný e-mail, potrebujeme na to jeho verejný kľúč. Čo zašifrujeme týmto verejným kľúčom, to bude viesť dešifrovať len ten, kto vlastní príslušný súkromný kľúč, a to je náš adresát. Verejný kľúč príjemcu sa získa z jeho osobného certifikátu, ktorý si najprv musíme do e-mailového klienta nainštalovať. Následne pri vytváraní novej správy určenej tomuto adresátovi zvolíme, že chceme obsah mailu zašifrovať. E-mailový klient nájde príslušný certifikát podľa zhody v e-mailovej adrese nášho adresáta s e-mailovou adresou uvedenou v certifikáte. Ak je certifikát v poriadku⁴, zoberie sa z neho verejný kľúč, ním sa e-mail zašifruje a odošle.

Tu je dôležité poznamenať, že hlavičky e-mailu sa nešifrujú, šifruje sa len jeho obsah. K e-mailu pribudnú nasledujúce hlavičky:

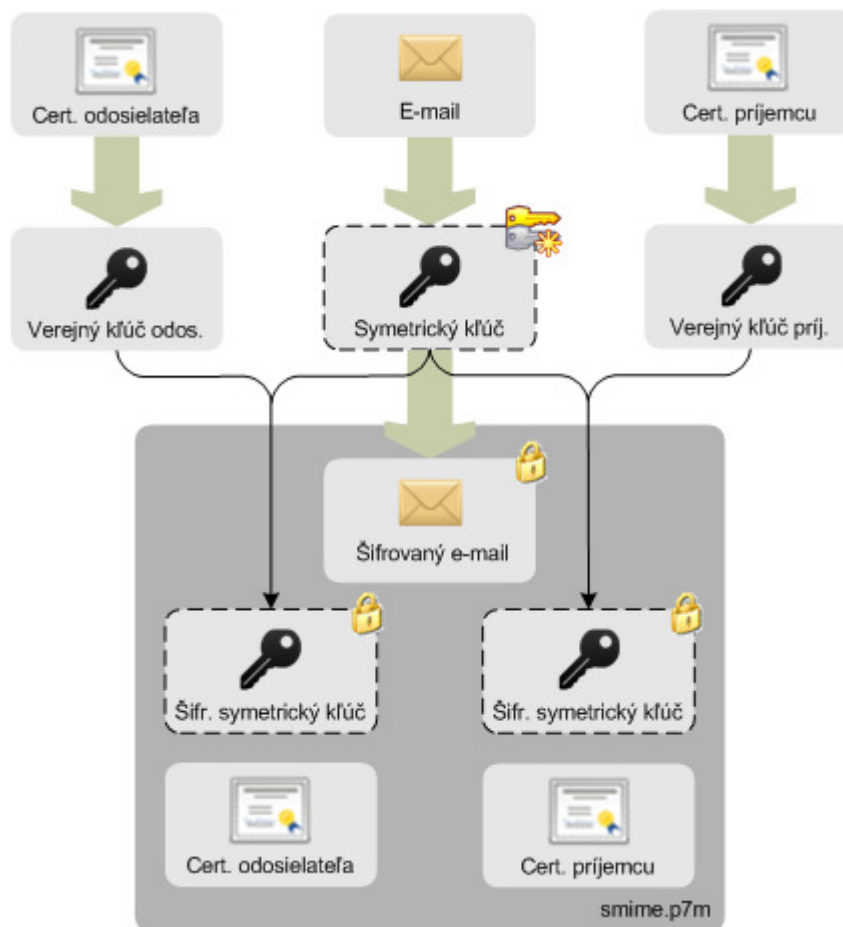
```
Content-Type: application/pkcs7-mime; name="smime.p7m"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7m"
Content-Description: S/MIME Encrypted Message
```

Je vidieť, že zašifrovaný e-mail predstavuje telo správy v kódovaní Base64 (viac o tomto kódovaní v časti 2.5.3). Klienti, ktorí nevedia šifrovanú správu spracovať, zobrazia súbor *smime.p7m* ako prílohu - súbor formátu PKCS#7. Mozilla Thunderbird okrem samotného zašifrovaného tela e-mailu pribalí do tohto súboru certifikát adresáta aj certifikát odosielateľa. Je to preto, aby obe strany vedeli, ktorý súkromný kľúč majú na dešifrovanie použiť (podľa toho, ktorý kľúč prináleží príslušnému certifikátu).

Obsah e-mailu sa kvôli rýchlosti nešifruje priamo verejným kľúčom adresáta. Namiesto toho sa vygeneruje jeden symetrický kľúč určený priamo na šifrovanie mailu. E-mail sa zašifruje týmto symetrickým kľúčom a tento kľúč sa zašifruje toľko krát, koľko je príjemcov, ich verejnými kľúčmi (každým raz). Proces šifrovania pre jedného príjemcu

⁴ napr. je platný, je vydaný dôveryhodnou certifikačnou autoritou, je určený na šifrovanie...

ilustruje obr. 2-5. Jedným z príjemcov je sám odosielateľ, pretože šifrovaný e-mail sa umiestni do jeho odoslanej pošty a mal by ho vedieť neskôr prečítať, ak by potreboval.



Obr. 2-5: Znáozornenie šifrovania e-mailu v programe Mozilla Thunderbird

Príjemca takéhoto šifrovaného e-mailu zistí podľa priloženého certifikátu, ktorý zo svojich súkromných kľúčov má použiť na dešifrovanie. Dešifruje najprv symetrický kľúč a ním dešifruje samotný e-mail.

Pri šifrovaní sa nepoužíva súkromný kľúč, jeho účasť je nutná až pri dešifrovaní.

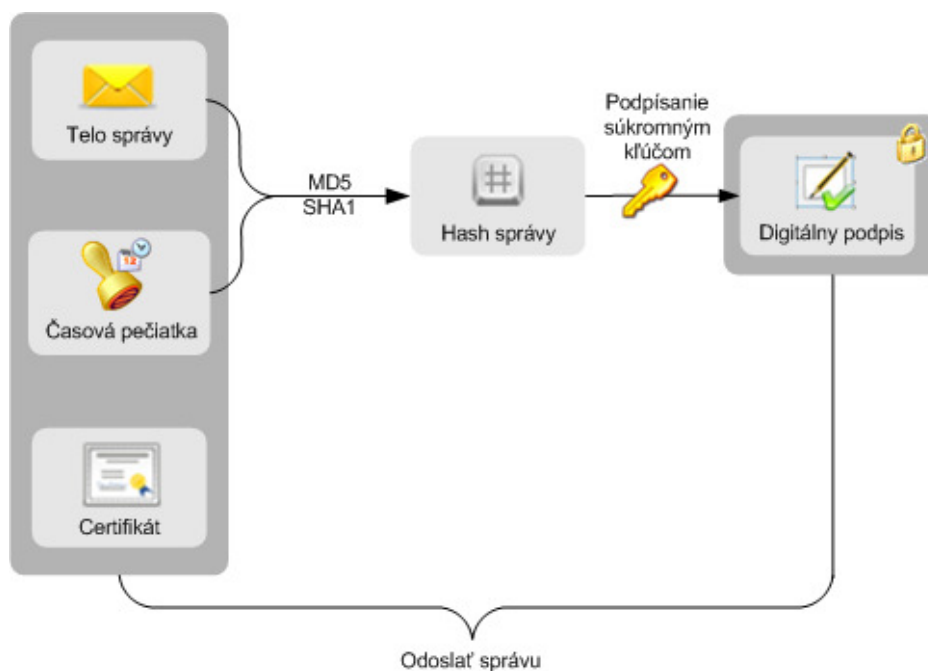
2.2.2 Digitálny podpis v praxi

Pri digitálnom podpisovaní e-mailu potrebujeme použiť náš súkromný kľúč. Pre pripomenutie, princíp digitálneho podpisu spočíva v tom, že hash nami vytvorenej správy sa zašifruje naším súkromným kľúčom. Mozilla Thunderbird takto vytvorený digitálny podpis zaobalí do súboru s názvom *smime.p7s*. Ten obsahuje okrem podpisu a časovej pečiatky aj informáciu, aký hashovací algoritmus bol použitý na získanie hashu správy a tiež obsahuje aj osobný certifikát odosielateľa kvôli tomu, aby príjemca dokázal overiť identitu odosielateľa a rovno mal k dispozícii aj jeho verejný kľúč.

Obsah e-mailu sa posielá v priamom texte a súbor *smime.p7s* je poslaný ako príloha. Ako vyžaduje štandard S/MIME, k e-mailu pribudne hlavička:

```
Content-Type: multipart/signed; protocol="application/pkcs7-signature";  
micalg=sha1; boundary="..."
```

Príjemca sa pozrie do súboru *smime.p7s* a extrahuje z neho certifikát odosielateľa. Zistí, či je tento certifikát v poriadku (či bol vydaný dôveryhodnou certifikačnou autoritou, či je ho možné použiť na podpisovanie správ, či je platný atď.). Ak je certifikát v poriadku, môže si ho uložiť do svojho úložiska osobných certifikátov a tým umožniť posielanie šifrovaných správ tejto osobe v budúcnosti. Potom príjemca zistí, aký hashovací algoritmus treba použiť na získanie hashu správy. Ten použije a výsledný hash porovná s hashom, ktorý bol získaný dešifrovaním podpisu. Ak sa zhodujú, podpis je platný.



Obr. 2-6: Znáznornenie digitálneho podpisovania e-mailu

2.3 KRYPTOGRAFICKÉ ZARIADENIA, TOKENY

Počítač, v ktorom máme v úmysle používať náš súkromný kľúč, je potenciálne nebezpečné prostredie, tým skôr, ak to nie je náš počítač. Nevieme, aké hrozby v ňom môžu číhať - v podobe spywaru⁵, keyloggeru⁶, bezpečnostných dier v softvéri... v skratke, nemali by sme mať v počítači umiestnený súkromný kľúč, aby nedošlo k jeho odcudzeniu takouto cestou. S tým súvisí práve myšlienka použitia špecializovaného hardvéru na uchovávanie citlivých informácií (kľúčov). Ten by vykonával kryptografické operácie, na ktoré sú vyžadované citlivé údaje, ako je súkromný kľúč.

- *Kryptografický token* - nosič citlivých údajov, ako je súkromný kľúč, zvyčajne má podobu čipovej karty. Zvykne obsahovať aj osobný certifikát a ďalšie certifikáty. Môže podporovať ukladanie aj iných objektov, ako sú verejné kľúče, symetrické kľúče a iné. Čip v tokene stráži súkromné údaje a poskytuje čítačke len to, čo uzná za vhodné.
- *Čítačka kryptografických tokenov* - hardvér pripojený zväčša pomocou rozhrania USB. Má jeden alebo viacero *slotov* - miest, kam sa vkladajú tokeny. Súčasťou ovládačov čítačky bývajú moduly PKCS#11 resp. CryptoAPI, ktoré poskytujú programom v PC jednotné rozhranie prístupu k tokenom.

Čítačky zvyknú mať displej a tlačidlá na zadanie PIN kódu na prístup k tokenu. Tie, ktoré toto nemajú, vyžadujú zadanie PIN kódu z počítača, kde je tento kód vystavený potenciálnemu nebezpečenstvu (obr. 1-1).

Mobilný telefón môžeme postaviť do role kryptografického tokenu a čítačky v jednom - je to určite bezpečnejšie prostredie, než počítač. Mobilný telefón má dnes každý a jeho použitie v tejto roli prináša vyššiu bezpečnosť pri nulových dodatočných nákladoch.

⁵ Spyware - škodlivý softvér určený na sledovanie akcií užívateľa pri práci s počítačom

⁶ Keylogger - spyware zaznamenávajúci stlačené klávesy

2.3.1 Rozhrania prístupu k tokenom

Výrobcov kryptografických zariadení je mnoho a preto bolo treba špecifikovať jednotné rozhranie na komunikáciu počítačových programov s týmito zariadeniami. Prístup k tomuto rozhraniu je štandardizovaný, čím sa abstrahuje od vnútornej implementácie zariadenia samotného. Poznáme dve takéto rozhrania: *PKCS#11* (iný názov: Cryptoki) a *CryptoAPI* (alebo skrátené CAPI). Porovnanie ich rozdielov ponúka tabuľka 2-1.

PKCS#11	CryptoAPI
Autor: RSA Laboratories	Autor: Microsoft
Programy používajú PKCS#11 moduly	Programy používajú CSP (Cryptographic Service Providers)
Multiplatformné	Dostupné len v MS Windows
Ponúka len najzákladnejšie funkcie	Umožňuje aj pokročilé funkcie
Rozšírenejšie	Menej rozšírené

Tab. 2-1: Porovnanie PKCS#11 a CryptoAPI.

CryptoAPI nedefinuje len rozhranie, aké majú mať CSP moduly, ale je to kompletné API umožňujúce rôzne kryptografické operácie. Príkladom môže byť, že CryptoAPI umožňuje overiť platnosť certifikátu (aj s jeho nadradenými certifikátmi), keďže má prístup k systémovému úložisku certifikátov. Na rozdiel od toho, PKCS#11 definuje len objekt typu „certifikát“ a aké má mať atribúty. Je to teda skutočne len rozhranie modulov na komunikáciu počítačových programov s kryptografickým zariadením, a všetky vyššie funkcie musí implementovať daný program. Ten môže mať vlastné úložisko certifikátov, alebo môže použiť to systémové (ako CryptoAPI).

Pre účely tejto diplomovej práce bolo zvolené rozhranie PKCS#11 práve kvôli jeho rozšírenosti. Toto rozhranie je používané programami ako Mozilla Firefox, Mozilla Thunderbird, Netscape Navigator, či Adobe Acrobat. Produkty Mozilly a Netscape používajú NSS (Network Security Services), čo je súbor knižníc na vykonávanie kryptografických operácií, prácu s PKCS#11 modulmi, spravovanie vlastných úložísk certifikátov a kľúčov, ich overovanie atď. Viac informácií o NSS možno nájsť na stránke <http://www.mozilla.org/projects/security/pki/nss/>.

2.4 KRYPTOGRAFICKÉ ŠTANDARDY

2.4.1 Súbor štandardov PKCS

Štandardy PKCS (Public-Key Cryptography Standards) sú špecifikácie vytvorené RSA Laboratories v spolupráci s autormi bezpečnostných systémov kvôli zrýchleniu nasadzovania kryptografie s verejnými kľúčmi [6]. Podľa [6] v súčasnosti existujú nasledujúce štandardy PKCS:

- PKCS#1: RSA Cryptography Standard (obsahuje PKCS#2 aj PKCS#4),
- PKCS#3: Diffie-Hellman Key Agreement Standard,
- PKCS #5: Password-Based Cryptography Standard,
- PKCS #6: Extended-Certificate Syntax Standard,
- PKCS #7: Cryptographic Message Syntax Standard,
- PKCS #8: Private-Key Information Syntax Standard,
- PKCS #9: Selected Attribute Types,
- PKCS #10: Certification Request Syntax Standard,
- PKCS #11: Cryptographic Token Interface Standard,
- PKCS #12: Personal Information Exchange Syntax Standard,
- PKCS #13: Elliptic Curve Cryptography Standard,
- PKCS #14: Pseudo Random Number Generation,
- PKCS #15: Cryptographic Token Information Format Standard.

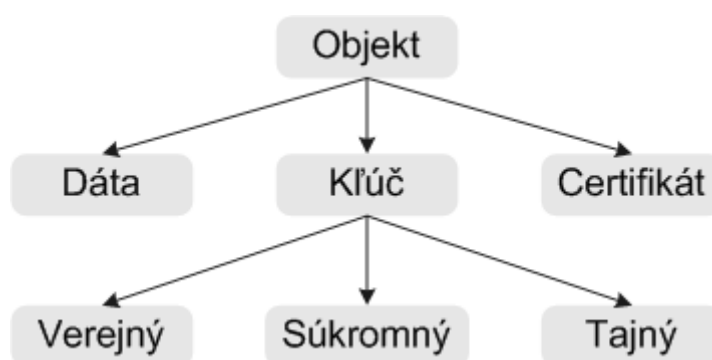
2.4.1.1 PKCS#11

Štandard PKCS#11 (alebo Cryptoki) určuje rozhranie kryptografických tokenov dostupné počítačovým programom, ktoré majú záujem využívať kryptografické funkcie týchto tokenov. Toto rozhranie je definované množinou funkcií, štruktúr, typov premenných atď. v hlavičkovom súbore jazyka C. Kryptografický modul býva predstavovaný knižnicou DLL (vo Windows) resp. SO (v Unixových systémoch), pričom aj statické linkovanie je možné. Volaním funkcií z tejto knižnice je zabezpečená komunikácia s tokenom z pohľadu počítačového programu. Tento modul však nemusí komunikovať ďalej, môže to byť čisto softvérový modul tváriaci sa ako fyzický token (produkty Mozilly majú takéto dva moduly vstavané). Podstatné je, že rozhranie komunikácie s programom je stanovené a čo sa deje za kryptografickým modulom, už dotýčný program nemusí zaujímať.

Táto kapitola poskytuje stručný pohľad aplikácie na token tak, ako je definované v PKCS#11 (podľa [7]).

Cryptoki poskytuje rozhranie k jednému alebo viacerým fyzickým zariadeniam tak, že ich reprezentuje pomocou *slotov*. Každý slot patriaci k fyzickému zariadeniu môže obsahovať *token*. Logická reprezentácia tokenu je v podobe zariadenia, ktoré uchováva *objekty* a môže vykonávať rôzne kryptografické operácie.

Objekty môžu byť troch typov: *dáta*, *klúč* a *certifikát*. Objekt typu *dáta* je všeobecný objekt definovaný aplikáciou. Objekt typu *klúč* obsahuje kryptografický klúč a objekt typu *certifikát* obsahuje certifikát. Klúč môže byť *súkromný*, *verejný* alebo *tajný* (symetrický). Toto rozdelenie objektov je znázornené na obrázku 2-7.



Obr. 2-7: Klasifikácia objektov v PKCS#11

Objekty sa tiež delia do dvoch skupín:

- *Objekty viažuce sa k tokenu (token objects)* - viditeľné pre všetky aplikácie prístupujúce k danému tokenu, a ostávajú uložené na tokene aj po tom, ako sú tieto aplikácie ukončené a token je vybratý zo slotu.
- *Objekty viažuce sa k relácii (session objects)* - sú dočasné v tom zmysle, že keď sa ukončí relácia, v ktorej boli vytvorené, tieto objekty zaniknú. Sú viditeľné len z tej aplikácie, ktorá ich vytvorila (keďže relácia sa viaže k aplikácii).

Ďalšie delenie objektov je podľa prístupu k nim:

- *Verejné objekty* - sú zobraziteľné aplikácii aj bez prihlásenia k tokenu.
- *Súkromné objekty* - na ich zobrazenie musí byť užívateľ autentifikovaný k danému tokenu (napr. pomocou PIN kódu).

Každý objekt má jeden alebo viac *atribútov*. Atribúty môžu byť všeobecné (napr. atribút udávajúci typ objektu) alebo špecifické, viažuce sa k určitému typu objektu (napr. verejný exponent RSA kľúča).

Cryptoki vyžaduje, aby aplikácia mala otvorenú aspoň jednu *reláciu* (angl. *session*) prv, než bude manipulovať s objektmi či volať niektoré funkcie. Relácia predstavuje logické spojenie medzi aplikáciou a tokenom a môže byť buď len na čítanie (R/O) alebo na čítanie i zápis (R/W) token objektov. Obe môžu čítať objekty viažuce sa k tokenu a čítať i zapisovať objekty viažuce sa k relácii, ale len R/W relácia môže zapisovať objekty viažuce sa k tokenu.

Token môže mať limitovaný počet R/O i R/W relácii, ktoré smú byť s ním otvorené súčasne. Otvorená relácia môže byť vo viacerých stavoch. Stav relácie určuje jej prístup k objektom a funkciám na manipuláciu s objektmi. Stavy sa menia s prihlásením a odhlásením užívateľa k tokenu. Tiež Cryptoki rozlišuje dva typy užívateľov - bežného užívateľa a superužívateľa. Detaily sú popísané v [7], ale keďže v tejto diplomovej práci sa táto forma autentifikácie nepoužíva, nie je nutné sa jej ďalej venovať.

Relácie aj objekty sú jednoznačne určené identifikátorom (angl. *handle*), ktorý je reprezentovaný typom *unsigned int*. Platné identifikátory musia mať vždy nenulovú hodnotu a používajú sa pri volaní funkcií, pri ktorých je potrebné sa odkázať na niektorý objekt alebo reláciu.

Citlivé objekty (ako napr. kľúče) môžu byť označené ako *sensitive* (nemôžu byť odhalené mimo tokenu) či *unextractable* (nemôžu byť odhalené mimo tokenu ani len v zašifrovanej podobe), ale stále ich možno použiť v rámci tokenu ako kľúče.

V Cryptoki sú jednotlivé kryptografické algoritmy reprezentované ako tzv. mechanizmy (*mechanisms*). Každý má svoj jednoznačný identifikátor, ktorý sa používa pri odkazovaní na mechanizmus napríklad pri volaní dešifrovacích funkcií.

PKCS#11 modul nemusí podporovať všetky funkcie definované týmto štandardom, stačí len určitá ich podmnožina; podľa toho, na čo je daný modul (i token) určený. Návratový typ funkcií je CK_RV (definovaný ako *unsigned int*) a používa sa na určenie výsledku vykonávania danej funkcie. Funkcia nemusí zlyhať, môže prebehnúť úspešne, a pritom jej návratová hodnota môže byť iná než CKR_OK. Kompletný zoznam funkcií podľa [7] je uvedený v prílohe 1. Funkcie nepodporované v implementácii tejto diplomovej práce majú v tabuľke preškrtnutý názov.

2.5 ALGORITMY

Táto kapitola sa zaoberá popisom najbežnejších algoritmov a spôsobov kódovaní obsahu používaných v kryptografickej praxi.

2.5.1 Kryptosystém RSA

RSA je asymetrický kryptosystém - používa dvojicu kľúčov, z ktorej jeden sa používa ako súkromný a druhý ako verejný. Čo zašifrujeme jedným kľúčom, môžeme odšifrovať len druhým z danej dvojice a naopak. RSA sa považuje za bezpečný kryptosystém, pokiaľ sú použité dostatočne dlhé kľúče (v súčasnosti sa za minimum považuje dĺžka kľúča 1024 bitov). Používa sa vo viacerých oblastiach, najmä v digitálnom podpisovaní a šifrovaní. Zdroj pre túto kapitolu je [8].

2.5.1.1 RSA - generovanie kľúčov

Postup generovania kľúčov je nasledujúci:

- Zvoliť dve prvočísla p a q - mali by byť zvolené náhodne a mali by mať približne rovnaký počet bitov.
- Vypočítať $n = p * q$ - tzv. *modulus* spoločný pre oba kľúče.
- Vypočítať $\varphi(n) = (p - 1) * (q - 1)$
- Zvoliť také e a d , že $0 < e < \varphi(n)$ aj $0 < d < \varphi(n)$ a aby platilo, že $e * d \equiv 1 \pmod{\varphi(n)}$. e sa nazýva „verejný exponent“ a najčastejšie sa používa číslo 65537. Číslo d sa nazýva „súkromný exponent“.
- Verejný kľúč je dvojica (e, n) a súkromný kľúč je dvojica (d, n) .

2.5.1.2 RSA - šifrovanie

Máme k dispozícii verejný kľúč osoby, ktorej chceme poslať správu x zašifrovanú na správu y . Šifrujeme výpočtom $y = x^e \pmod{n}$. Správu y odošleme adresátovi.

2.5.1.3 RSA - dešifrovanie

Dostali sme správu y zašifrovanú naším verejným kľúčom. Dešifrovanú správu x získame výpočtom: $x = y^d \pmod{n}$.

Z dôvodu vyššej efektívnosti mnohé knižnice (Java, .NET, OpenSSL) vypočítavajú aj ďalšie hodnoty, ktoré sa ukladajú ako súčasti súkromného kľúča:

- $d_p = d \bmod (p - 1)$
- $d_q = d \bmod (q - 1)$
- $q_{inv} = q^{-1} \bmod p$

2.5.2 Hashovacie algoritmy

Pri digitálnom podpisovaní je nutné zo správy vytvoriť jej odtlačok (tzv. hash) a ten zašifrovať. Nemalo by zmysel šifrovať celú správu, lebo podpis by bol taký dlhý, ako správa samotná. Hash je oproti správe malý (a tiež má vždy rovnakú dĺžku) a teda aj podpis bude malý. Existuje mnoho spôsobov, ako zo správy vytvoriť jej odtlačok. Hashovacie funkcie musia mať nasledujúce vlastnosti [9]:

- pre každú správu je ľahké vypočítať jej hash,
- pre každý hash je ťažké nájsť takú správu, ktorá by mala tento hash,
- pre každú správu je ťažké nájsť inú správu s rovnakým hashom,
- je ťažké nájsť dve rôzne správy s rovnakým hashom.

Všeobecný postup tvorby hashu správy je nasledujúci [9]:

- správa sa rozdelí na rovnako dlhé bloky m_1, m_2, \dots, m_k ,
- každý hashovací algoritmus má pevne stanovený inicializačný vektor IV , teda položí sa $h_0 = IV$,
- rekurzívne sa počíta $h_i = f(m_i, h_{i-1})$ pre $i = 1$ až k ,
- výsledný odtlačok správy je h_k .

Keďže správa sa rozdelí na rovnako dlhé bloky, na jej koniec zvyčajne treba pridať potrebný počet bitov, aby posledný blok mal požadovanú dĺžku. Spôsob takéhoto ukončenia (angl. *padding*) správy závisí od hashovacieho algoritmu.

V súčasnosti najpoužívanjšie sú hashovacie funkcie MD5 a SHA-1.

- MD5 dáva 128-bitový hash a spracúva bloky o dĺžke 512 bitov,
- SHA-1 vracia 160-bitový hash a spracúva tiež 512-bitové bloky.

2.5.3 Kódovanie Base64

Podľa [10], Base64 sú kódovacie schémy používané na reprezentáciu binárnych dát textovými dátami. Base64 sa používa, ak je treba preniesť binárne dáta kanálom, ktorý je navrhnutý na prenos textových dát. Používa sa aj pri potrebe ukladania binárnych dát do textového súboru, napríklad do XML súboru. Bežne sa používa napríklad na kódovanie príloh k e-mailom, kódovanie zašifrovaného a podpísaného e-mailu, a tiež certifikáty i kľúče zvyknú byť uložené nie v ich binárnej podobe, ale zakódované pomocou Base64 (viac v časti 2.6).

Množina znakov povolených v Base64 sa líši v rôznych implementáciách. Základnou požiadavkou je, aby tieto znaky boli spoločné pre čo najviac kódovaní textu (ASCII, UTF-8 a ďalšie) a hlavne aby sa dali vypísať (angl. *printable*). Najbežnejšia (a v e-mailoch používaná) implementácia má názov MIME Base64. Povoľuje znaky A-Z, a-z, 0-9, + (plus) a / (lomka), pričom ich indexy sú určené v tomto poradí, od 0 do 63. Ako znak na doplnenie dĺžky výsledného reťazca na násobok 4 bajtov, sa používa = (rovná sa).

Prevod zdrojových dát do Base64 je jednoduchý a deje sa po 3-bajtových blokoch:

- prevod každého bajtu do jeho binárnej reprezentácie,
- spojenie výsledných bitov za sebou do reťazca dlhého 24 bitov,
- rozdelenie tohto 24-bitového reťazca na štyri 6-bitové bloky,
- každý 6-bitový blok môže mať jednu zo 64 rôznych hodnôt,
- prevod každého 6-bitového bloku na jeden znak z Base64.

Príklad pre prvý blok textu „Diplomová práca“:

Bajt textu	D								i								p							
ASCII kód	68								105								112							
Bity	0	1	0	0	0	1	0	0	0	1	1	0	1	0	0	1	0	1	1	1	0	0	0	0
Base64 index	17								6								37							
Base64 znak	R								G								l							

Tab. 2-2: Príklad kódovania Base64

Celý text „Diplomová práca“ sa zakóduje na „RGlwbG9tb3bDoSBwcsOhY2E=“.

2.5.4 ASN.1 a DER kódovanie

Podľa [11], ASN.1 (*Abstract Syntax Notation One*) je štandard flexibilnej notácie pre reprezentáciu dát. Je to štandard ISO/IEC a ITU-T, pôvodne definovaný v roku 1984. V roku 1988 sa presunul pod štandard X.208 a v roku 1995 pod X.680. Najnovšia verzia pochádza z roku 2002.

ASN.1 definuje abstraktnú syntax dát, ale neobmedzuje spôsob ich kódovania. Existujú rôzne pravidlá kódovania dát reprezentovaných pomocou ASN.1:

- Basic Encoding Rules (BER)
- Distinguished Encoding Rules (DER)
- Canonical Encoding Rules (CER)
- XML Encoding Rules (XER)
- Packed Encoding Rules (PER)
- Generic String Encoding Rules (GSER)

ASN.1 spolu so špecifickými pravidlami kódovania umožňuje výmenu dát medzi rôznymi systémami nezávisle od architektúry počítačov a jazyka implementácie. Je mnoho oblastí, kde sa používa ASN.1 spolu s niektorým jeho kódovaním, ale momentálne je najdôležitejšie, že sa hojne používa v kryptografickej praxi. Certifikáty a kľúče sú uložené v DER kódovaní, čo je „podmnožina“ BER kódovania.

Príklad z [11]:

Majme definovaný dotaz a odpoveď pre protokol „FooProtocol“ v notácii ASN.1 (toto je spôsob jeho definície autormi protokolu):

```
FooProtocol DEFINITIONS ::= BEGIN

    FooQuestion ::= SEQUENCE {
        trackingNumber INTEGER,
        question      IA5String
    }

    FooAnswer ::= SEQUENCE {
        questionNumber INTEGER,
        answer          BOOLEAN
    }

END
```

Majme potom nejakú správu tohto protokolu, typu FooQuestion. Tá môže vyzerat' v ASN.1 napríklad takto:

```
myQuestion FooQuestion ::= {  
    trackingNumber      5,  
    question            "Anybody there?"  
}
```

Aby bolo možné takúto správu poslať cez počítačovú sieť, potrebujeme ju nejako reprezentovať v postupnosti bitov. Ako to urobiť, definujú pravidlá kódovania dát v ASN.1, najjednoduchšie z nich je DER (Distinguished Encoding Rules). Špecifikácia protokolu musí priamo uvádzať, ktoré konkrétne kódovanie sa používa, aby obe komunikujúce strany vedeli, že budú používať DER. Hore uvedená správa by v DER vyzerala takto (v hexadecimálnej reprezentácii bajtov):

```
30 13 02 01 05 16 0e 41 6e 79 62 6f 64 79 20 74 68 65 72 65 3f
```

Vysvetlenie bajt po bajte:

- 30 - značka pre sekvenciu (SEQUENCE)
- 13 - dĺžka sekvencie v bajtoch
- 02 - značka pre celé číslo (INTEGER)
- 01 - dĺžka čísla v bajtoch
- 05 - hodnota
- 16 - značka pre IA5String (čo je obyčajný ASCII reťazec)
- 0e - dĺžka reťazca
- 41 6e 79 62 6f 64 79 20 74 68 65 72 65 3f - hodnota („Anybody there?“ v ASCII)

DER kódovanie používa trojice „typ - dĺžka - hodnota“, čo umožňuje čítanie a interpretáciu prenášaného obsahu už počas príjmu. Nevýhodou z pohľadu zhoršenej čitateľnosti pre človeka je binárna reprezentácia. Samozrejme, DER kódovanie je omnoho zložitejšie, než je ukázané na tomto príklade. Napríklad ak je nejaká hodnota dlhšia ako 255 bajtov, používa sa na reprezentáciu tejto dĺžky iná notácia. Podrobnosti a ďalšie príklady sú do hlbšej miery vysvetlené v [12].

2.6 FORMÁTY

2.6.1 Formáty certifikátov

Podľa PKCS#6 [13] je štruktúra X.509 certifikátu v ASN.1 daná takto:

```
Certificate ::= SEQUENCE {  
    certificateInfo CertificateInfo,  
    signatureAlgorithm AlgorithmIdentifier,  
    signature BIT STRING }
```

kde *CertificateInfo* má nasledujúcu štruktúru:

```
CertificateInfo ::= SEQUENCE {  
    version [0] Version DEFAULT v1988,  
    serialNumber CertificateSerialNumber,  
    signature AlgorithmIdentifier,  
    issuer Name,  
    validity Validity,  
    subject Name,  
    subjectPublicKeyInfo SubjectPublicKeyInfo }
```

```
Version ::= INTEGER { v1988(0) }
```

```
CertificateSerialNumber ::= INTEGER
```

```
Validity ::= SEQUENCE {  
    notBefore UTCTime,  
    notAfter UTCTime }
```

```
SubjectPublicKeyInfo ::= SEQUENCE {  
    algorithm AlgorithmIdentifier,  
    subjectPublicKey BIT STRING }
```

```
AlgorithmIdentifier ::= SEQUENCE {  
    algorithm OBJECT IDENTIFIER,  
    parameters ANY DEFINED BY ALGORITHM OPTIONAL }
```

Štruktúra *CertificateInfo* je kódovaná v DER a podpísaná algoritmom, ktorého identifikátor udáva *signatureAlgorithm*. Podpis je uložený v *signature*. Celý certifikát je tiež kódovaný v DER. Štruktúra *CertificateInfo* môže obsahovať aj ďalšie, voliteľné polia, ako je napr. webová adresa certifikačnej autority.

Poznáme dva základné spôsoby, ktorými certifikát môže byť uložený do súboru. Ak je len v „surovom“ DER formáte, súbor má príponu *.der*, zriedkavo *.cer*. Druhý spôsob

je PEM formát: to znamená, že DER formát sa zakóduje do Base64 a výsledný reťazec sa zariadkuje vždy po 64 znakoch. Pred tieto riadky sa vloží riadok -----BEGIN CERTIFICATE----- a na koniec sa vloží riadok -----END CERTIFICATE----- . Takto uložené certifikáty majú príponu .crt, .pem alebo .cer. Keďže prípona .cer sa používa pre oba typy certifikátov, jediný spôsob ich rozlíšenia je skontrolovaním ich obsahu. Certifikáty druhého typu môžu byť v jednom súbore aj viaceré za sebou, ale neodporúča sa to, pretože programy môžu mať so spracovaním takýchto certifikátov problémy.

Implementácia tejto diplomovej práce podporuje oba typy certifikátov (prípony .der a .crt), pričom v jednom súbore môže byť len jeden certifikát.

2.6.2 Formáty súkromných kľúčov

Syntax v ASN.1 pre súkromné kľúče je popísaná v štandarde PKCS#8 [14]. Súkromný kľúč môže byť šifrovaný alebo nešifrovaný. Oba typy súkromných kľúčov popisuje tento štandard. Nešifrovaný súkromný kľúč má nasledujúce vyjadrenie:

```
PrivateKeyInfo ::= SEQUENCE {  
    version Version,  
    privateKeyAlgorithm PrivateKeyAlgorithmIdentifier,  
    privateKey PrivateKey,  
    attributes [0] IMPLICIT Attributes OPTIONAL }  
  
Version ::= INTEGER  
  
PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier  
  
PrivateKey ::= OCTET STRING  
  
Attributes ::= SET OF Attribute
```

Štruktúra *PrivateKey* je rôzna v závislosti od algoritmu kľúča. Napríklad pre súkromný kľúč RSA je to štruktúra *RSAPrivateKey* kódovaná v BER.

Šifrovaný súkromný kľúč vyzerá podľa PKCS#8 v ASN.1 takto:

```
EncryptedPrivateKeyInfo ::= SEQUENCE {  
    encryptionAlgorithm EncryptionAlgorithmIdentifier,  
    encryptedData EncryptedData }  
  
EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier  
  
EncryptedData ::= OCTET STRING
```

Nešifrovaný súkromný kľúč je najskôr vyjadrený v DER a výsledok je zašifrovaný symetrickým kľúčom. Zašifrované dáta sa umiestnia do položky *encryptedData* a identifikátor šifrovacieho algoritmu, ktorý bol použitý na toto šifrovanie, sa umiestni do položky *encryptionAlgorithm*.

Často sa používa formát šifrovaného súkromného kľúča s *Proc-Type* a *DEK-Info* (tento formát je aj predvolený v OpenSSL). Ten vyzerá takto:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: DES-EDE3-CBC, 37D929A5D17AA5FE

Q1a3hAQcyFxFW02EbtzROdzPnOr4/0BsAncF21GD57Ewgay1+wAAokBXiPxOuKzBO
...
-----END RSA PRIVATE KEY-----
```

Riadok s *Proc-Type* značí, že nasledujúca PEM správa (kľúč v Base64) bola zašifrovaná. V riadku s *DEK-Info* je uvedené, aký algoritmus a v akom režime bol na šifrovanie použitý. Tiež sa tu uvádza inicializačný vektor tohto algoritmu. V tomto prípade bol použitý 3DES v režime CBC (Cipher Block Chaining).

Všetky typy súkromných kľúčov bývajú uložené v PEM formáte do súboru s príponou *.pem*, kde sú zakódované v Base64, podobne ako certifikáty. Rozdiel je v prvom a poslednom riadku súboru. Prehľadnejšie je to zobrazené v tabuľke 2-3.

	Prvý riadok	Posledný riadok
Certifikát	-----BEGIN CERTIFICATE-----	-----END CERTIFICATE-----
Nešifr. PKCS#8 kľúč	-----BEGIN RSA PRIVATE KEY-----	-----END RSA PRIVATE KEY-----
Šifr. PKCS#8 kľúč	-----BEGIN ENCRYPTED PRIVATE KEY-----	-----END ENCRYPTED PRIVATE KEY-----
Šifr. kľúč s Proc-Type a DEK-Info	-----BEGIN RSA PRIVATE KEY-----	-----END RSA PRIVATE KEY-----
Verejný kľúč	-----BEGIN PUBLIC KEY-----	-----END PUBLIC KEY-----

Tab. 2-3: Prvý a posledný riadok kľúčov a certifikátov v Base64

Implementácia tejto diplomovej práce podporuje nešifrované súkromné kľúče formátu PKCS#8 a šifrované kľúče s *Proc-Type* a *DEK-Info* šifrované pomocou 3DES v režime CBC (pričom kľúče tohto typu aj sama vytvára).

2.6.3 Formáty verejných kľúčov

Verejný kľúč je vyjadrený v nasledujúcej ASN.1 štruktúre:

```
PublicKeyInfo ::= SEQUENCE {  
    algorithm AlgorithmIdentifier,  
    PublicKey BIT STRING  
}
```

Pole *PublicKey* je závislé od hodnoty identifikátora algoritmu. Napríklad pre verejný kľúč RSA je to vnorená štruktúra *RSAPublicKey* definovaná takto:

```
RSAPublicKey ::= SEQUENCE {  
    modulus INTEGER,  
    publicExponent INTEGER  
}
```

Toto všetko je zakódované v DER a uložené do súboru v PEM formáte tak, že dáta v DER formáte sú zakódované ešte do Base64 a pribudne prvý a posledný riadok podľa tabuľky 2-3. Súbor s takto uloženým verejným kľúčom má príponu *.pem*. Takéto kľúče sú podporované programovým vybavením vytvoreným v rámci tejto diplomovej práce.

3 ANALÝZA A NÁVRH RIEŠENIA

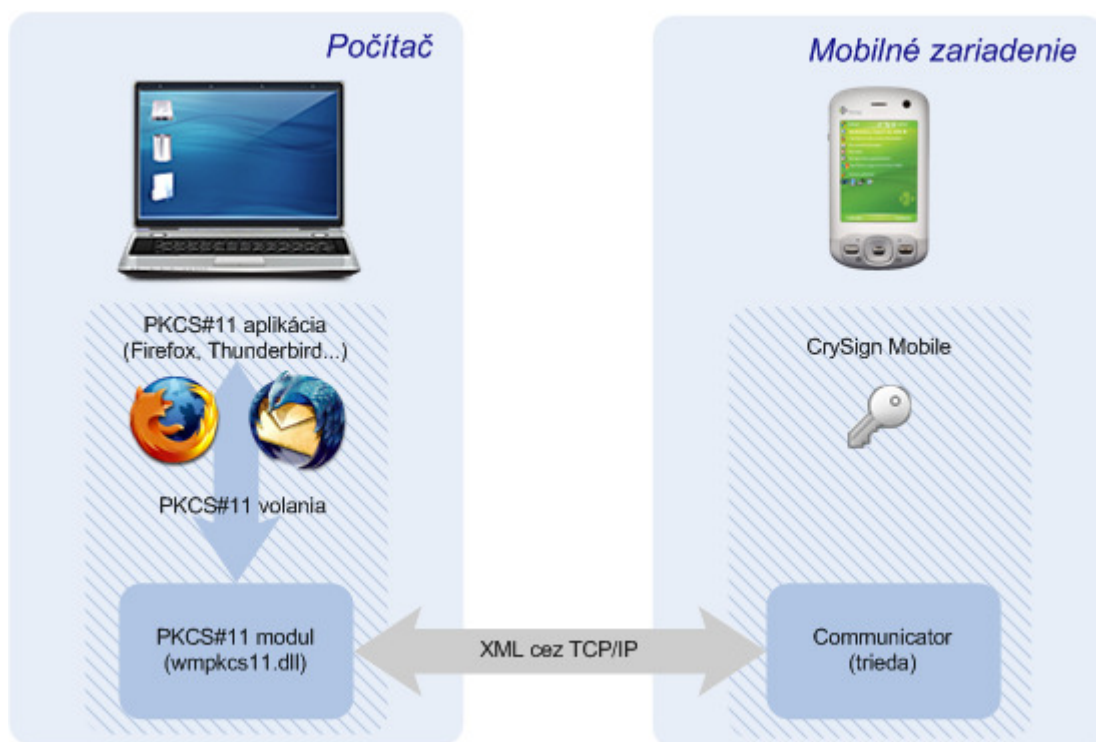
Myšlienka použitia bežného mobilného zariadenia v úlohe kryptografického tokenu nie je nová. Už v roku 1999 sa podarilo výskumnému tímu z *Princeton University* implementovať PKCS#11 modul a softvérové vybavenie pre mobilné zariadenie (PDA) 3Com PalmPilot [15]. Tento softvér úspešne použili pri digitálnom podpise a dešifrovaní e-mailov a tiež sa im podarilo implementovať generovanie RSA kľúčov. Keďže hardvér spomínaného PDA bol značne pomalý, kryptografické operácie ani zďaleka nedosahovali rýchlosť pri použití bežnej čipovej karty. Napríklad, podpísanie 1024-bitovým RSA kľúčom trvalo 25 sekúnd, generovanie RSA kľúčov o tejto dĺžke dokonca 30 minút!

To bol zároveň posledný (autorovi tejto práce známy) pokus o realizáciu takejto myšlienky, a aj preto bolo vhodné pokúsiť sa o to v rámci diplomovej práce s modernejšími prostriedkami. Táto kapitola sa snaží ukázať celkový návrh riešenia takéhoto programového vybavenia, a to predstretím spôsobu rozdelenia aplikácií i komunikácie medzi nimi.

3.1 CELKOVÝ POHĽAD NA RIEŠENIE

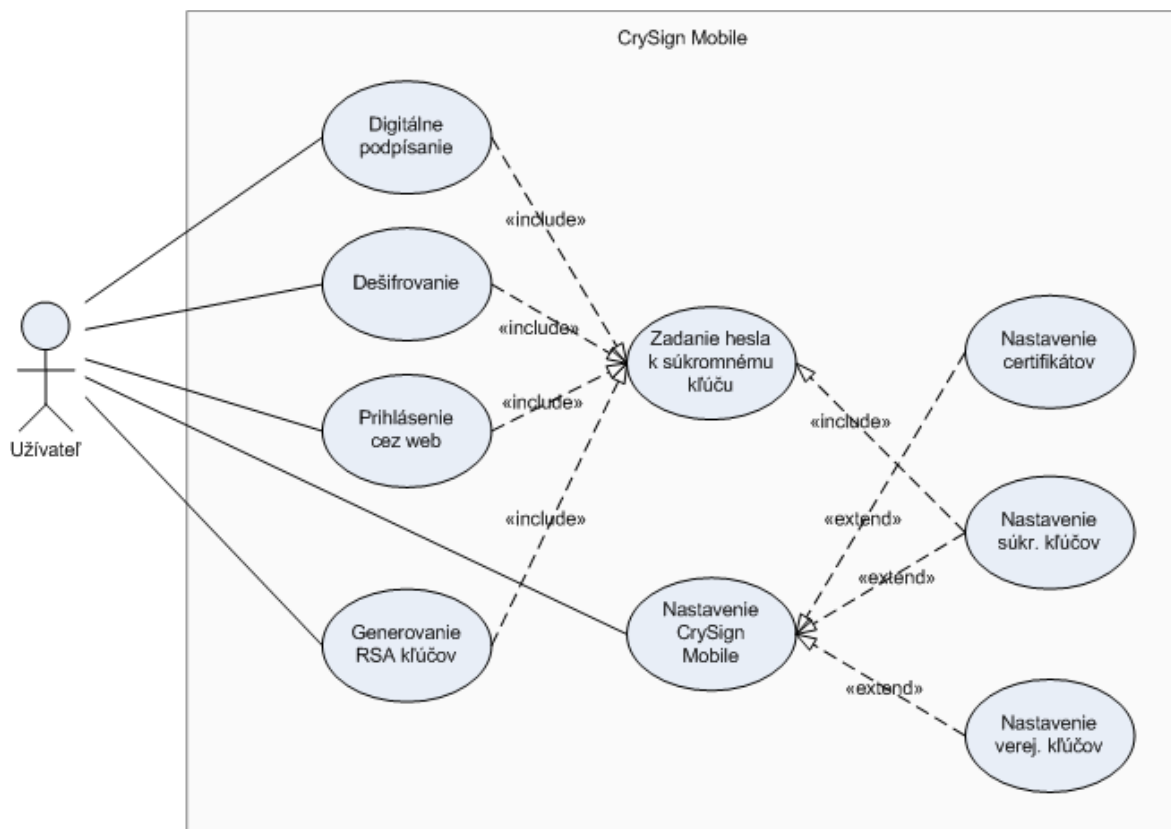
Softvérové vybavenie vytvorené pre účely tejto diplomovej práce dostalo názov *CrySign* (*Crypto* + *Sign* - šifrovanie a podpisovanie). Skladá sa z dvoch častí - softvér na strane PC je predstavovaný PKCS#11 modulom načítaným v programe, ktorý ovláda štandard PKCS#11 a je teda schopný používať jeho služby. Softvér na strane mobilného zariadenia (resp. telefónu) je program na tomto zariadení bežiaci a komunikujúci so softvérom na strane PC (*CrySign Mobile*). Všetky kryptografické operácie sa vykonávajú v mobilnom zariadení a PKCS#11 modul iba sprostredkováva požiadavky aplikácie z počítača tomuto zariadeniu, a preberá a poskytuje odpoveď z mobilného zariadenia tejto aplikácii. Takýto návrh umožňuje zmeniť platformu telefónu za nejakú inú (napr. z Windows Mobile na Android, ak by bol softvér *CrySign Mobile* pre Android dostupný) a to bez nutnosti ďalšej práce na zmenách PKCS#11 modulu.

Obrázok 3-1 znázorňuje rozloženie tohto softvéru pri jeho použití. Na obrázku 3-2 sa nachádza diagram prípadov použitia softvéru *CrySign Mobile*.



Obr. 3-1: Celkový pohľad na softvér CrySign

V počítači beží nejaký PKCS#11 program, napríklad Mozilla Thunderbird. Tento program ponúka možnosť načítať PKCS#11 modul, ktorý sa bude používať pre kryptografické operácie (samozrejme, takýchto modulov môže byť aj viac). PKCS#11 program komunikuje s týmto modulom tak, že volá jeho funkcie definované v štandarde PKCS#11 (príloha 1). Modul komunikuje s mobilným zariadením pomocou TCP/IP a na aplikačnej vrstve používa XML (detaily v kapitole 3.2). V mobilnom zariadení beží softvér *CrySign Mobile* a komunikáciu má v ňom na starosti trieda *Communicator*. Tento softvér poskytuje kryptografické operácie počítaču a pri tom v prípade potreby interpretuje kľúče a certifikáty, ktoré v ňom boli nakonfigurované. Ďalšie detaily softvéru *CrySign Mobile* nie sú na obrázku zobrazené, ale podrobnosti sú uvedené v kapitole 4. Rovnako, ako funguje PKCS#11 modul, je popísané v kapitole 5.



Obr. 3-2: Diagram prípadov použitia softvéru CrySign Mobile

Použitie softvéru *CrySign* umožňuje v počítači pomocou rozhrania modulu PKCS#11 vykonávať v mobilnom zariadení nasledujúce operácie:

- digitálny podpis,
- dešifrovanie,
- prihlasovanie sa na webové stránky pomocou osobného certifikátu,
- generovanie RSA kľúčov,
- ... a pri tom všetkom súkromné kľúče nikdy neopustia prostredie telefónu.

Keďže sú súkromné kľúče v telefóne šifrované, každé použitie takéhoto kľúča vyžaduje zadanie hesla k nemu. Počas generovania kľúčov je požadované heslo použité na zašifrovanie novo vygenerovaného súkromného kľúča.

3.2 NÁVRH KOMUNIKÁCIE

PKCS#11 modul s mobilným zariadením musí nejakou formou komunikovať. Zo všetkých možností sa ako najlepšia ukázala komunikácia pomocou TCP/IP (na sieťovej vrstve) kvôli univerzálnosti takejto komunikácie. Zoberme si najbežnejšie spôsoby, pomocou ktorých môže mobilné zariadenie byť pripojené k počítaču:

- kábel (alebo kolíska),
- Wi-Fi,
- Bluetooth,
- infračervený prenos.

Vo väčšine prípadov existuje softvér (či už v telefóne, alebo v PC), ktorý umožňuje nad týmito pripojeniami vytvoriť tzv. PAN⁷. V takejto sieti sa väčšinou používa protokol TCP/IP. Na druhej strane, pokiaľ je telefón pripojený cez Wi-Fi, môže to byť v tzv. ad-hoc režime (bez prístupového bodu, priamo k PC), alebo môže byť pripojený k nejakému prístupovému bodu. Pokiaľ je tento prístupový bod rovnaký, ako prístupový bod, ku ktorému je pripojený počítač, navonok v podstate nie je rozdiel medzi PAN a klasickou počítačovou sieťou bežiacou na Wi-Fi.

Ďalšou z výhod protokolu TCP/IP je jeho rozšírenosť v použití bežných počítačových sietí a internetu. To znamená, že telefón nemusí ani byť v rovnakej IP sieti, ako počítač, dokonca môže byť „na druhej strane zemegule“. Stačí, keď je na príslušnom TCP porte dostupný z počítača, kde chceme jeho služby používať a poznáme jeho verejnú IP adresu, ktorú si nastavíme v konfiguračnom súbore PKCS#11 modulu na počítači.

V prípadoch väčšiny zariadení s mobilným operačným systémom Windows sa pri ich prvom pripojení k počítaču nainštalujú ovládače a podporný softvér od Microsoftu s názvom *ActiveSync* (alebo po novom *Sync Center*). Slúži na synchronizáciu údajov z telefónu, ale predovšetkým vytvorí v počítači virtuálny sieťový adaptér s IP adresou 169.254.2.2/16. Telefón má potom IP adresu 169.254.2.1/16, pričom tieto adresy sú vždy rovnaké. Keď takáto sieť existuje medzi počítačom a telefónom, možno medzi nimi ľubovoľne komunikovať prostredníctvom TCP/IP protokolu.

Samozrejme, PKCS#11 modul sa nemôže spoľahnúť, že na adrese 169.254.2.1 nájde vždy mobilný telefón. Riešenie tohto problému je popísané v kapitole 5.2, ktorá sa zaoberá softvérom na strane PC - PKCS#11 modulom.

⁷ PAN (Personal Area Network) - počítačová sieť na komunikáciu zariadení v bezprostrednej blízkosti.

CrySign Mobile načúva na dvoch portoch:

- TCP/57404 - tu prebieha bežná komunikácia s PC pomocou výmeny XML dokumentov (popísaná v tejto kapitole),
- UDP/57404 - tento port slúži na odpovedanie PKCS#11 modulu pri vyhľadávaní dostupných mobilných zariadení (viac v kapitole 5).

Komunikácia prebieha pomocou výmeny XML dokumentov, čo má výhodu pomerne jednoduchého spracovania na oboch stranách. Štruktúrou týchto XML dokumentov je určené rozhranie PKCS#11 modulu aj softvéru na strane telefónu, čo ponúka možnosť abstrakcie jedného od druhého - pri dodržaní tohto rozhrania môžeme zameniť jednu časť bez zmeny druhej.

Mobilné zariadenie je vždy v úlohe servera a PKCS#11 modul v úlohe klienta. XML dokumenty smerujúce od PKCS#11 modulu majú nasledujúcu štruktúru:

```
<?xml version="1.0" encoding="Windows-1250" ?>
<xmlinfo>
    <request type="(1)" application="(2)">
        (3)
    </request>
</xmlinfo>
```

kde:

- (1) určuje typ požiadavky, resp. volanie určitej PKCS#11 funkcie. Príkladom môže byť reťazec „GetAttributeValue“.
- (2) predstavuje identifikátor PKCS#11 aplikácie. Je to prvých 8 znakov z MD5 hashu ID procesu v systéme a systémového času prvej správy, ktorá bola zaslaná mobilnému zariadeniu.
- (3) obsahuje parametre požiadavky, zvyčajne sú to ďalšie vnorené XML značky. Ich názvy a obsah závisí od toho, aký typ požiadavky (1) tento XML dokument obsahuje.

Odpoveď od softvéru na strane telefónu má takýto formát:

```
<?xml version="1.0" encoding="Windows-1250" ?>
<xmlinfo>
    <response to="(1)">
        (2)
    </response>
</xmlinfo>
```

- (1) udáva typ odpovede - má rovnakú hodnotu, ako atribút *type* (1) v požiadavke prislúchajúcej k tejto odpovedi.
- (2) obsahuje parametre odpovede (vnorené XML značky) a závisí od parametrov požiadavky, ktorá tejto odpovedi predchádzala.

V oboch prípadoch dochádza často k potrebe prenášať binárne dáta. Keďže XML je textový formát, na tento účel sa binárne dáta najprv zakódujú pomocou Base64 (popísané v kapitole 2.5.3). Odpoveď v značke *response* máva aj značku *result*, ktorá udáva celkový výsledok vykonanej PKCS#11 funkcie - jej návratovú hodnotu.

Príklad na funkcii *FindObjects* - požiadavka:

```
<?xml version="1.0" encoding="Windows-1250" ?>
<xmlinfo>
  <request type="FindObjects" application="b3776f56">
    <handle>1</handle>
    <maxcount>10</maxcount>
    <objectlist>?</objectlist>
    <objectcount>?</objectcount>
  </request>
</xmlinfo>
```

Odpoveď na túto požiadavku vyzerá napríklad takto:

```
<?xml version="1.0" encoding="Windows-1250" ?>
<xmlinfo>
  <response to="FindObjects">
    <objectlist>
      <object>1</object>
    </objectlist>
    <objectcount>1</objectcount>
    <result>0</result>
  </response>
</xmlinfo>
```

Funkcia *FindObjects* spustí hľadanie objektov podľa vzoru, ktorý bol uvedený v predchádzajúcom volaní *FindObjectsInit*. V uvedenom príklade PKCS#11 aplikácia požaduje spustenie tejto operácie nad reláciou s číslom 1 a maximálny počet vrátených objektov má byť 10. Značkami obsahujúcimi otázniky sa pýta na ich obsah, ktorý je v odpovedi naplnený (výstupné parametre PKCS#11 funkcie). Odpoveď vráti jeden objekt, ktorý vyhovoval vzoru a ten má číslo 1. Výsledok - návratová hodnota - celej funkcie je CKR_OK, čo predstavuje hodnota 0.

3.3 POUŽITÉ PRODUKTY TRETÍCH STRÁN

V programátorskej praxi často človek narazí na problém, na ktorý samotný programovací jazyk neposkytuje riešenie, ale ktorý už pred ním niekto vyriešil. Takýchto riešení môže byť viacero, líšia sa možnosťami, ktoré podporujú, rýchlosťou, zložitou použitím a pod. Aj v tejto diplomovej práci bolo treba na niektorých miestach použiť už hotové riešenia a táto kapitola sa zaoberá popisom, kde a prečo sú použité.

3.3.1 Použitý kód tretích strán na strane PC

3.3.1.1 TinyXML

TinyXML je jednoduchý a malý XML parser napísaný v C++ kóde nezávislom na operačnom systéme. Je použitý v mnohých open-source či komerčných aplikáciách. V tejto diplomovej práci sa používa na čítanie odpovede z mobilného zariadenia na strane PKCS#11 modulu, keďže samotný jazyk C++ funkcie na prácu s XML dokumentmi neobsahuje. TinyXML pozostáva zo súborov *tinyst.h*, *tinyxml.h*, *tinyst.cpp*, *tinyxml.cpp*, *tinyxmlerror.cpp* a *tinyxmlparser.cpp*. Webová stránka tohto projektu je na adrese <http://www.grinninglizard.com/tinyxml> a autorom je Lee Thomason.

3.3.1.2 Base64 kódovač a dekodovač

Funkcie pre prácu s textom kódovaným v Base64 boli prebraté z webovej stránky <http://www.adp-gmbh.ch/cpp/common/base64.html>, autorom je René Nyffenegger. Tieto funkcie sa používajú pri prenose binárnych dát pomocou XML súborov. Bolo treba pridať jednu kódovaciu funkciu tak, aby sa okrem reťazcov typu *unsigned char const ** dal kódovať aj vstupný reťazec *std::string*. Kód týchto funkcií sa nachádza v súboroch *base64.h* a *base64.cpp*.

3.3.1.3 MD5

Autorom použitej implementácie hashovacej funkcie MD5 je Gary McNickle a kód je v súboroch *md5.h* a *md5.cpp*. Používa sa pri generovaní identifikátora aplikácie pre použitie v XML požiadavkách na mobilný telefón. Aj tu bolo treba spraviť malé úpravy - tie spočívali v zamenení „potenciálne nebezpečných“ funkcií za ich bezpečné varianty tak, aby sa kód skompiloval vo Visual Studiu 2008 bez varovaní. Funkcia *fopen* bola zamenená za *fopen_s*, *sprintf* za *sprintf_s* a *strncat* za *strncat_s*.

3.3.2 Použitý kód tretích strán na strane mobilného telefónu

3.3.2.1 Funkcie pre vyvolanie natívnych CryptoAPI funkcií

V .NET Compact Frameworku bohužiaľ nie sú všetky funkcie, ktoré ponúka operačný systém - v tomto prípade chýbali funkcie pre digitálny podpis hashu zloženého z výsledkov hashovacích funkcií SHA-1 a MD5. Tento podpis sa používa pri prihlasovaní na webové stránky pomocou osobného certifikátu. Funkcie, ktoré sa starajú o tzv. „marshalling“ do a z API funkcií operačného systému, sa nachádzajú v projekte *PInvokeLib* v súbore *WinCrypto.cs*. Boli prevzaté z webovej stránky MSDN: <http://blogs.msdn.com/b/alejacula/archive/2007/11/23/p-invoking-cryptoapi-in-net-c-version.aspx>.

3.3.2.2 Detekcia výrobcu a platformy mobilného zariadenia

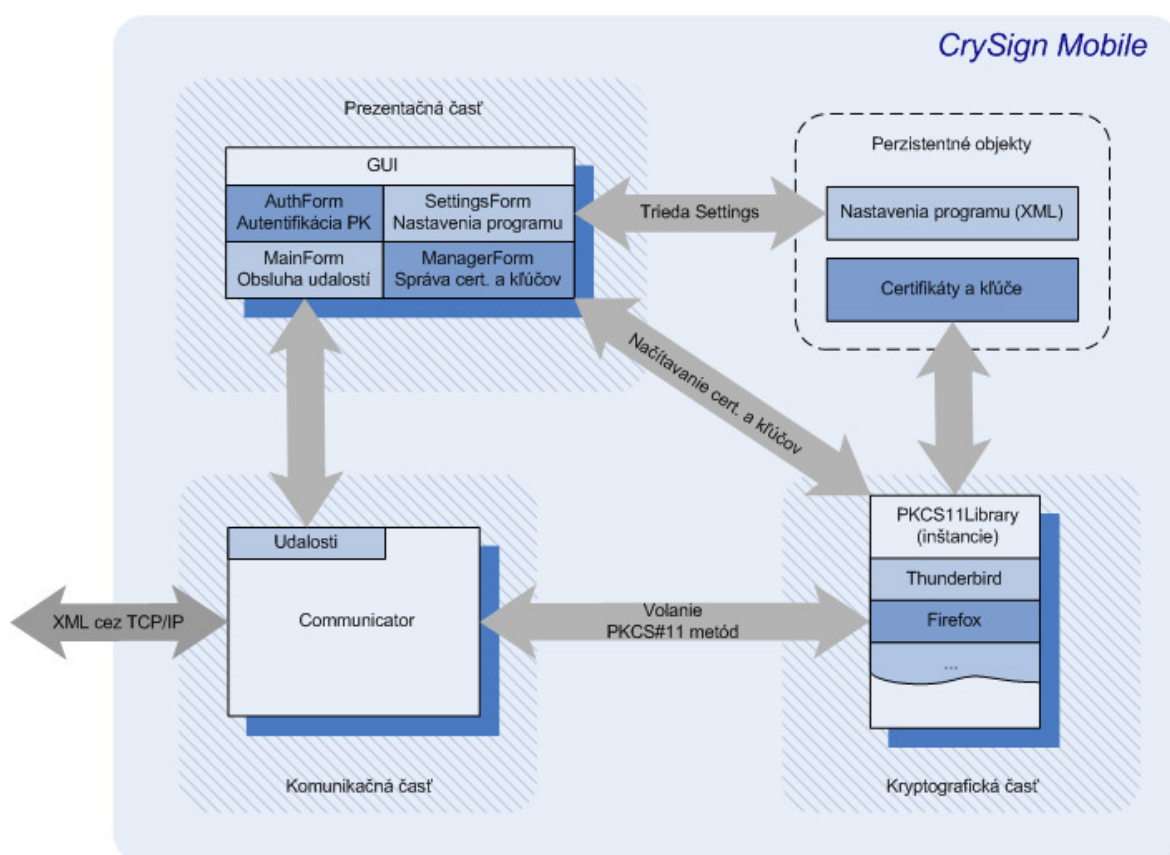
Keďže metódy na splnenie tejto úlohy nie sú opäť v .NET Compact Frameworku dostupné, bolo treba vyvolávať natívne funkcie operačného systému. Kód, pomocou ktorého sa to podarilo implementovať, pochádza z webovej stránky MSDN: <http://blogs.msdn.com/b/netcftteam/archive/2006/09/15/756755.aspx>. Tieto funkcie sa používajú na pomenovanie zariadenia a pod týmto menom sa zariadenie prezentuje PKCS#11 aplikácii. Nachádzajú sa v projekte *PInvokeLib* a v súbore *PInvoke.cs*.

3.3.2.3 Metódy na prácu s kľúčmi

Často bolo treba pracovať s kľúčmi v ich „surovej“, binárnej podobe tak, ako sa používajú v OpenSSL. Samotný .NET Compact Framework takéto funkcie neposkytuje, no riešenie je v kóde zverejnenom na stránke <http://www.jensign.com/opensslkey> (jeho autorom je JavaScience Consulting). Je tam program umožňujúci načítanie a spracovanie všemožných typov kľúčov a certifikátov. Z neho boli vybraté tie funkcie, ktoré našli uplatnenie v tejto diplomovej práci: *DecryptKey*, *EncryptKey*, *DecodeOpenSSLPrivateKey*, *DecodeOpenSSLPublicKey*, *DecodeRSAPrivateKey*, *DecodeX509PublicKey*, a k nim pomocné funkcie *GetIntegerSize* a *CompareByteArrays*. Tieto funkcie sa nachádzajú v projekte *PKCS11Lib* v súbore *Asn1.cs* v príslušne označenom regióne kódu.

4 SOFTVÉR NA STRANE TELEFÓNU

Softvér na strane telefónu - nazývaný *CrySign Mobile* - je v úlohe servera pri komunikácii s PKCS#11 modulom na strane PC (ten je v úlohe klienta). Načúva na portoch TCP i UDP číslo 57404, ako bolo spomenuté v kapitole 3.2. Na obrázku 4-1 je uvedená bloková schéma tohto softvéru.



Obr. 4-1: Bloková schéma softvéru CrySign Mobile

V princípe možno tento softvér rozdeliť do štyroch častí: komunikačná, kryptografická, prezentačná a perzistenčná.

Komunikačná časť je predstavovaná triedou *Communicator*, ktorá obsahuje vlákna majúce na starosti načúvanie na portoch TCP a UDP. Tieto spracúvajú komunikáciu na týchto portoch prebiehajúcu, a ak nastane nejaká udalosť, pri ktorej treba notifikovať prezentačnú časť (trebárs kvôli zmene v GUI), tak sa táto udalosť vyvolá a o jej konkrétnu implementáciu sa postará metóda prezentačnej časti na túto udalosť zavesená. Pokiaľ si táto udalosť vyžaduje volanie nejakej PKCS#11 funkcie, tak sa zavolá príslušná metóda z kryptografickej časti a jej výsledok sa spracuje na strane komunikačnej časti do XML súboru (odpovede) a táto sa odošle PKCS#11 modulu.

V kryptografickej časti, ktorá obsahuje inštancie triedy *PKCS11Library*, sa pre každú PKCS#11 aplikáciu vytvorí jedna inštancia (konkrétne pri volaní PKCS#11 funkcie *C_Initialize*) a po jej ukončení zase táto inštancia zanikne (volanie *C_Finalize*). Každá takáto inštancia obsahuje stav pre danú aplikáciu a tým pádom sa jednotlivé aplikácie medzi sebou neovplyvňujú.

Načítavanie kľúčov a certifikátov umiestnených v úložisku telefónu sa deje pomocou statických metód triedy *PKCS11Library*. K perzistenčnej časti okrem týchto súborov patrí aj súbor s nastaveniami aplikácie s názvom *crysign.xml*, s ktorým manipuluje statická trieda *Settings* a táto manipulácia sa deje priamo z formulára na to určeného.

4.1 POPIS TRIED A METÓD

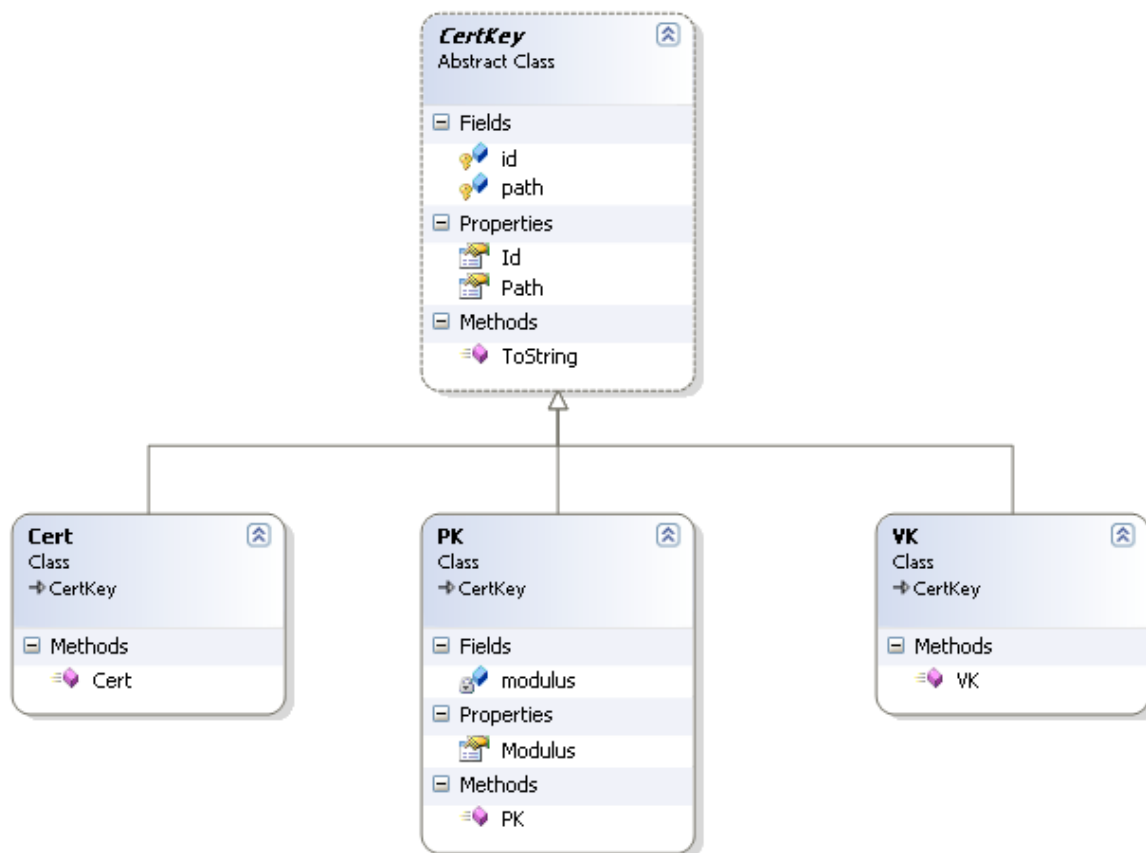
Softvér *CrySign Mobile* sa (vo VS2008) skladá z nasledujúcich projektov:

- CrySign Mobile (aplikácia),
- PKCS11Lib (knižnica) - PKCS#11 knižnica na strane telefónu, kryptografická časť,
- CertKeyLib (knižnica) - knižnica s triedami reprezentujúcimi certifikáty a kľúče,
- PInvokeLib (knižnica) - obsahuje triedy na volanie funkcií operačného systému,
- XMLBuildLib (knižnica) - slúži na zostavenie odpovede vo formáte XML,
- Setup (inštalačný balík).

Toto rozdelenie na projekty bolo nutné jednak kvôli oddeleniu aplikačnej logiky od implementácie PKCS#11 funkcií, a jednak kvôli zdieľaniu niektorých tried medzi týmito projektmi (tak napr. vznikol projekt *XMLBuildLib*).

Z dôvodu nedostatku miesta sú na tomto mieste v texte uvedené len diagramy tried projektov *CertKeyLib* a *XMLBuildLib*. Ostatné sú uvedené v prílohách 2, 3 a 4. Ku každému z uvedených projektov nasleduje v tejto kapitole popis jeho tried a metód.

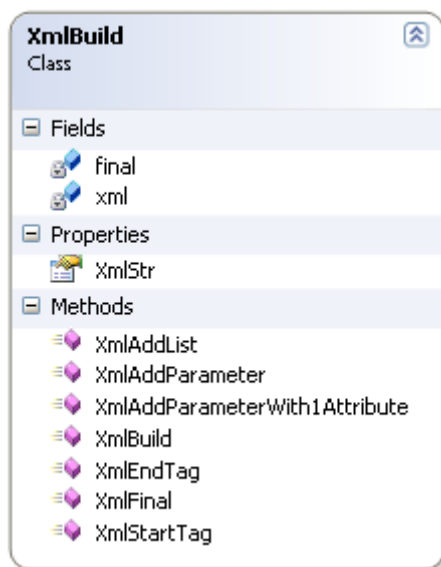
4.1.1 Projekt CertKeyLib



Obr. 4-2: Diagram tried projektu *CertKeyLib*

V projekte *CertKeyLib* sa nachádzajú triedy reprezentujúce certifikáty a kľúče (*PK* - súkromný kľúč, *VK* - verejný kľúč). Všetky tri majú spoločného predka - abstraktnú triedu *CertKey* - a to preto, že certifikáty aj kľúče potrebujú mať definovaný identifikátor a cestu k nim v súborovom systéme telefónu. Súkromný kľúč uchováva aj modulus, ktorý je spoločný s príslušným verejným kľúčom, avšak PKCS#11 aplikácie sa zvyknú pýtať na modulus súkromného kľúča, a preto by sme ho mali vedieť poskytnúť bez predchádzajúceho dešifrovania tohto kľúča. Identifikátor kľúčov a certifikátov (atribút *id*) je číslo nastaviteľné v užívateľskom rozhraní aplikácie a viaže k sebe prislúchajúce certifikáty a kľúče. Tiež sa používa ako hodnota PKCS#11 atribútu *CKA_ID* príslušných PKCS#11 objektov, pričom tento atribút sa podľa [7] odporúča práve na tento účel.

4.1.2 Projekt XMLBuildLib



Obr. 4-3: Diagram tried projektu *XMLBuildLib*

Projekt *XMLBuildLib* a jeho trieda *XmlBuild* sa používa na vybudovanie odpovede v XML formáte, ktorá sa odošle PKCS#11 modulu na strane PC. Pri volaní konštruktora sa do súkromného atribútu *xml* umiestni hlavička odpovede a potom možno volať ďalšie metódy tvoriace telo odpovede - výsledného XML dokumentu. Tieto ďalšie metódy umožňujú pridávať ďalšie elementy do vnútra elementu *response*. Volaním metódy *XmlFinal* sa element *response* ukončí a potom sa ukončí aj celý XML dokument tak, že už ďalšie zmeny v ňom nemožno robiť.

4.1.3 Projekt CrySign Mobile

Tento projekt obsahuje aplikačnú logiku a užívateľské rozhranie softvéru *CrySign Mobile* a jeho diagram tried sa z dôvodu rozsiahlosti nachádza v prílohe 2. Projekt obsahuje štyri formuláre: hlavný formulár *MainForm*, formulár s nastaveniami aplikácie *SettingsForm*, formulár s nastaveniami kľúčov a certifikátov *ManagerForm* a formulár slúžiaci na vypývanie hesla k súkromnému kľúču od užívateľa, s názvom *AuthForm*.

Hlavnou triedou je trieda *Communicator*, ktorá má na starosti komunikáciu s PKCS#11 modulom na strane PC. Jej dôležitými metódami sú *ListeningThread* a *ListeningThreadUdp*, ktoré bežia v separátnych vláknach a zabezpečujú načúvanie a komunikáciu na TCP a UDP portoch. Tieto vlákna používajú na vyvolávanie udalostí delegátov tak, že títo delegáti odštartujú príslušné metódy, ktoré majú v názve *_EventFire* a v tele týchto metód sa vyvolá príslušná udalosť. Tieto metódy majú rovnaký predpis, ako

delegáti. Niekedy je treba okrem vyvolania udalosti pri jej vyvolaní preniesť nejaké dáta. To sa deje pomocou parametra typu *EventArgs* posúvaného vyvolaním udalosti. Pre tento účel sú tu triedy *IEventArgs* a *PassPhraseEventArgs* zaobľujúce príslušné dáta, ktoré treba pri vyvolaní udalosti preniesť do metód zavesených na tieto udalosti. Trieda *IEventArgs* sa používa pri prenose IP adresy klienta (počítača, na ktorom beží PKCS#11 modul) pri vyvolaní udalosti, keď tento modul nájde mobilné zariadenie pomocou UDP broadcastov. Trieda *PassPhraseEventArgs* obsahuje údaje potrebné pre zobrazenie formulára požadujúceho autentifikáciu užívateľa pre dešifrovanie súkromného kľúča - názov súboru s týmto kľúčom, heslo k tomuto kľúču, ktoré užívateľ zadal, a jeho voľbu, či si želá toto heslo zapamätať.

Trieda *Communicator* obsahuje odkaz na text z formulára (atribút *formLabel*), ktorý sa používa na vyvolávanie metód vo vlákne formulára pomocou metódy *Invoke*. Ďalej obsahuje zoznam (resp. slovník) inštancií PKCS#11 knižníc (triedy *PKCS11Library* z projektu *PKCS11Lib*) a tento zoznam obsahuje pre každý identifikátor PKCS#11 aplikácie jednu takúto inštanciu. Volajú sa vždy metódy tej inštancie PKCS#11 knižnice, ktorá má rovnaký identifikátor, ako bol uvedený v prichádzajúcom XML dokumente.

Ďalej je tu statická trieda *CertKeys*, ktorá obsahuje zoznamy práve načítaných kľúčov a certifikátov. Tieto zoznamy sa menia nielen pri nastavovaní užívateľom použitím formulára *ManagerForm*, ale aj pri generovaní kľúčov. Aby sa novo vygenerovaným kľúčom pridelil rovnaký identifikátor (keďže k sebe patria), sú tu pomocné atribúty *privKeyAdded* a *privKeyId*.

Statická trieda *Settings* slúži na spravovanie nastavení programu, keďže .NET Compact Framework na rozdiel od .NET Frameworku takúto možnosť nemá vstavanú. Táto trieda obsahuje súkromné atribúty spolu s vlastnosťami pre nastavenia programu a okrem toho má vlastnosti týkajúce sa kľúčov a certifikátov (ktoré odkazujú na príslušné vlastnosti triedy *CertKeys*). Množina vlastností triedy *Settings* v podstate predstavuje množinu dát, ktoré sa ukladajú do konfiguračného súboru s názvom *crysign.xml*. Metóda *Init* načíta tento súbor a naplní príslušné vlastnosti údajmi z neho, metóda *Save* uloží nastavenia programu do tohto súboru.

Súbor *crysign.xml* môže vyzeráť napr. takto:

```
<?xml version="1.0" encoding="Windows-1250" ?>
<settings>
    <ConfirmExit>0</ConfirmExit>
    <DebugEnabled>1</DebugEnabled>
```

```

<Mute>0</Mute>
<Cert id="1">\My Documents\Personal\Matej Kurpel.crt</Cert>
<PK id="1" modulus="pswts7EzOurJlBn9j39SEXv6ZmoBK...">\My
Documents\Personal\privkey_crypted.pem</PK>
<VK id="1">\My Documents\Personal\pubkey.pem</VK>
</settings>

```

V uvedenom príklade je načítaný certifikát a k nemu súkromný i verejný kľúč (sú prepojené pomocou identifikátora 1). K súkromnému kľúču sa ukladá modulus v binárnej podobe zakódovaný pomocou Base64 (v ukážke je skrátенý).

4.1.4 Projekt PKCS11Lib

Tento projekt predstavuje PKCS#11 knižnicu na strane telefónu a obsahuje všetko, čo je potrebné na dodržanie PKCS#11 špecifikácie. Ponúka triedy reprezentujúce PKCS#11 objekty a ich atribúty i identifikátory (angl. *handle*), relácie, tokeny, sloty a mechanizmy. Všetky triedy sú zobrazené v diagrame tried tohto projektu (príloha 3).

Samotnú knižnicu predstavuje trieda *PKCS11Library*. K nej sa viaže množina objektov, relácií, slotov a zapamätaných hesiel k súkromným kľúčom, a tieto má ako svoje atribúty. Obsahuje prevažne metódy určené špecifikáciou PKCS#11, ktorých názov sa začína *C_*. Ďalej obsahuje statické metódy slúžiace na rôzne režijné úlohy súvisiace s vykonávaním PKCS#11 funkcií, napríklad načítanie kľúčov a certifikátov, či metódy riešiace zapamätávanie hesiel k súkromným kľúčom.

Zoznamy (objektov, mechanizmov, slotov, atribútov, relácií) sú reprezentované separátnymi triedami, ktoré zaobalujú generický *List<...>* príslušného typu, implementujú nad ním indexer a ponúkajú metódu na pridanie nového prvku do tohto zoznamu. Tieto triedy majú názvy *PKCS11ObjectList*, *MechanismList*, *SlotList*, *AttributeList* a *SessionList*.

PKCS#11 objekt je reprezentovaný triedou *PKCS11Object*; má svoj identifikátor (angl. *handle*) typu *Handle* a zoznam atribútov typu *AttributeList*. Trieda *AttributeList* implementuje rozhranie *IEquatable<AttributeList>*, ale metóda *Equals* neporovnáva dve inštancie na úplnú zhodu, avšak len na zhodu všetkých atribútov (*Attribute*) jednej inštancie s niektorými atribútmi druhej inštancie (tak, ako sa to vykonáva v PKCS#11 funkcii *C_FindObjects*).

PKCS#11 knižnica má množinu slotov typu *SlotList*, ktorá obsahuje prvky typu *Slot*. Tie okrem atribútov definovaných v špecifikácii PKCS#11 majú atribút *Token*, ktorý nie je *null* práve vtedy, keď v danom slotе je vložený token (v našom prípade je vždy token v slotе, pričom slot je práve jeden).

Slot má množinu podporovaných mechanizmov typu *MechanismList*. Mechanizmus typu *Mechanism* má len atribúty, ktoré určuje špecifikácia PKCS#11.

K PKCS#11 knižnici sa ešte viaže množina práve otvorených relácií reprezentovaná atribútom typu *SessionList*. Keďže operácie podpisovania, dešifrovania a vyhľadávania objektov sa viažu k relácii, trieda *Session* reláciu predstavujúca obsahuje atribúty a vlastnosti umožňujúce vykonávať tieto operácie. Relácia sa viaže k určitému slotu, takže je tu ešte odkaz na tento slot - využíva sa, keď PKCS#11 aplikácia požaduje ukončenie všetkých relácií otvorených na zadanom slotе.

Statická trieda *Asn1* obsahuje metódy potrebné pre manipuláciu s binárnymi dátami, ktoré predstavujú certifikáty a kľúče, a tiež podporné metódy na prácu s týmito súbormi uloženými v PEM formáte.

Posledná trieda má názov *PKCS11Const* a obsahuje definície konštánt z hlavičkového súboru PKCS#11, udáva sa v nej napríklad, že CKR_OK = 0.

4.1.5 Projekt PInvokeLib

Projekt *PInvokeLib* je nadstavbou nad natívnymi funkciami operačného systému, ktoré nie sú v .NET Compact Frameworku dostupné. Diagram tried je uvedený v prílohe 4. Statická trieda *PInvoke* ponúka možnosti zistenia názvu zariadenia a jeho výrobcu, spolu s typom platformy. Tieto údaje sa používajú na pomenovanie zariadenia voči PC. Tiež sú tu metódy na zapnutie a vypnutie displeja zariadenia - to sa hodí, keď je displej vypnutý a odrazu je treba vypýtať si od užívateľa heslo k súkromnému kľúču (a zvuk na tento fakt upozorňujúci môže byť vypnutý, resp. si ho užívateľ nemusí všimnúť).

Statická trieda *WinCrypto* je prevzatá z MSDN (podrobnosti v kapitole 3.3.2.1) a ponúka možnosti vyvolania kryptografických funkcií operačného systému, ktoré sú potrebné pri digitálnom podpise dát počas prihlasovania na webovú lokalitu pomocou súkromného kľúča.

4.2 SPRÁVA CERTIFIKÁTOV A KĽÚČOV

Certifikáty a kľúče sú uložené v mobilnom telefóne v súboroch rôzneho formátu. *CrySign Mobile* podporuje tieto formáty (podľa kapitoly 2.6):

- certifikáty v DER aj PEM formáte (*.der* a *.crt*),
- súkromné kľúče v PEM formáte (*.pem*) - nešifrované a šifrované s *DEK-Info* a *Proc-Type* (algoritmus DES-EDE3-CBC),
- verejné kľúče v PEM formáte (*.pem*).

Certifikáty a kľúče možno načítavať do programu z jeho hlavného menu - položka *Certs + Keys Manager*. Pri každom je zoznam aktuálne načítaných položiek v tvare „ID: názov súboru“. Pomocou ID sa certifikáty a kľúče viažu k sebe a pre aktuálne pridávanú položku (kľúč, certifikát), je možné zvoliť akékoľvek ID okrem už použitého.

Verejné kľúče sa v rámci aplikácie nikde nepoužívajú, ale možnosť ich načítania je dostupná pre budúce využitie a tiež sa rovno načíta verejný kľúč, ktorý bol vygenerovaný v rámci generovania kľúčových párov. K verejnému kľúču sa vytvorí PKCS#11 objekt s atribútom typu CKA_CLASS hodnoty CKO_PUBLIC_KEY. Hodnoty ďalších atribútov, ako napr. modulus (CKA_MODULUS) sa získajú použitím inštancie objektu *RSAParameters*, ktorý poskytuje trieda *RSACryptoServiceProvider* (obe triedy sú súčasťou .NET Compact Frameworku).

Pri pokuse načítať súkromný kľúč sa najprv zistí, či je tento kľúč šifrovaný, alebo nie. Ak nie je, aplikácia si vypýta heslo od užívateľa a toto heslo použije na zašifrovanie kľúča. Výsledný zašifrovaný kľúč je v PEM formáte s *DEK-Info* a *Proc-Type*. Ak je načítavaný kľúč už šifrovaný, overí sa jeho formát. Ak je v správnom formáte (šifrovaný pomocou DES-EDE3-CBC), užívateľ je požiadaný o heslo k tomuto kľúču kvôli načítaniu jeho modulusu. Modulus sa odloží do príslušného atribútu inštancie triedy *PK*, aby mohol byť poskytnutý, keď si ho PKCS#11 aplikácia vypýta. V prípade, že načítavaný kľúč nie je v správnom formáte (alebo bol pokus o načítanie verejného kľúča namiesto súkromného, či naopak), užívateľ je o tom informovaný chybovým hlásením a kľúč sa do zoznamu načítaných kľúčov nepridá. PKCS#11 objekt reprezentujúci súkromný kľúč má hodnotu atribútu CKA_CLASS = CKO_PRIVATE_KEY.

Ako už bolo spomenuté, je umožnené načítavanie dvoch typov certifikátov - PEM formátu alebo DER formátu. Typ certifikátu sa zisťuje pri jeho načítaní a ak je neznámy, nenačíta sa. Spracovanie X.509 certifikátov uľahčuje trieda *X509Certificate*, ktorá je

súčasťou .NET Compact Frameworku. Žiaľ, nie všetky potrebné funkcie sú v nej dostupné (alebo nemajú požadovaný výstup), preto bolo potrebné ich implementovať ručne (v triede *Asn1*). Napríklad, špecifikácia PKCS#11 požaduje atribút certifikátu typu CKA_ISSUER v DER formáte, ale metódy triedy *X509Certificate* ponúkajú len reprezentáciu v tvare reťazca. Preto bolo treba jeho reprezentáciu v DER kódovaní extrahovať z celého certifikátu vo formáte DER. Ďalším príkladom môže byť sériové číslo certifikátu, ktoré trieda *X509Certificate* poskytuje v obrátenej forme oproti tej, ktorá sa nachádza v certifikáte (a výsledok treba ešte zakódovať v DER). Certifikáty sú PKCS#11 objekty s hodnotou atribútu typu CKA_CLASS rovnou CKO_CERTIFICATE.

5 SOFTVÉR NA STRANE PC

Softvér na strane PC implementovaný pre účely tejto diplomovej práce je predstavovaný PKCS#11 modulom. Tento modul je DLL súbor, ktorý si môže aplikácia ovládajúca rozhranie PKCS#11 načítať do svojho pamäťového priestoru a volaním funkcií tohto modulu zabezpečovať kryptografické operácie s využitím zariadenia, ktoré sa k tomuto modulu viaže.

Rozhranie tohto modulu je dané špecifikáciou PKCS#11, ktorej súčasťou sú aj hlavičkové súbory pre implementáciu v jazyku C. Keďže špecifikácia je daná takto, nebolo tu možné ostať „v manažovanom svete“, ale namiesto toho bolo nutné implementovať PKCS#11 modul v jazyku C++.

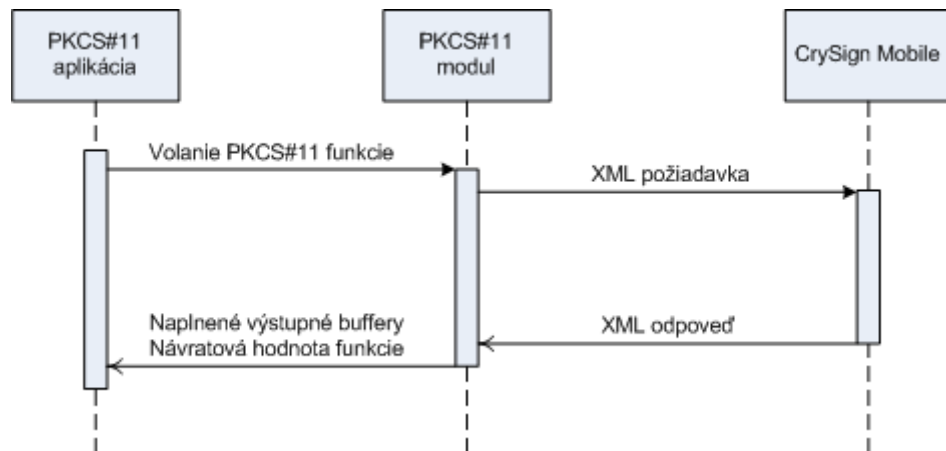
Táto kapitola sa zaoberá popisom fungovania PKCS#11 modulu a to najprv z pohľadu fungovania väčšiny funkcií (ktoré je podobné) a potom aj z celkového pohľadu na následnosť volania funkcií. V ďalšom nasleduje popis spôsobu, ako PKCS#11 modul nájde zariadenie vo svojom okolí, a nakoniec ako možno toto hľadanie prispôbiť svojim požiadavkám použitím konfiguračného súboru PKCS#11 modulu.

5.1 VŠEOBECNÝ POHĽAD NA PKCS#11 FUNKCIE

Keď aplikácia využívajúca PKCS#11 modul zavolá niektorú z jeho funkcií, tento modul potrebuje komunikovať s mobilným telefónom, odpoveď spracovať a sprostredkovať ju aplikácii. Preto tieto funkcie vyzerajú veľmi podobne a princíp ich fungovania je nasledujúci:

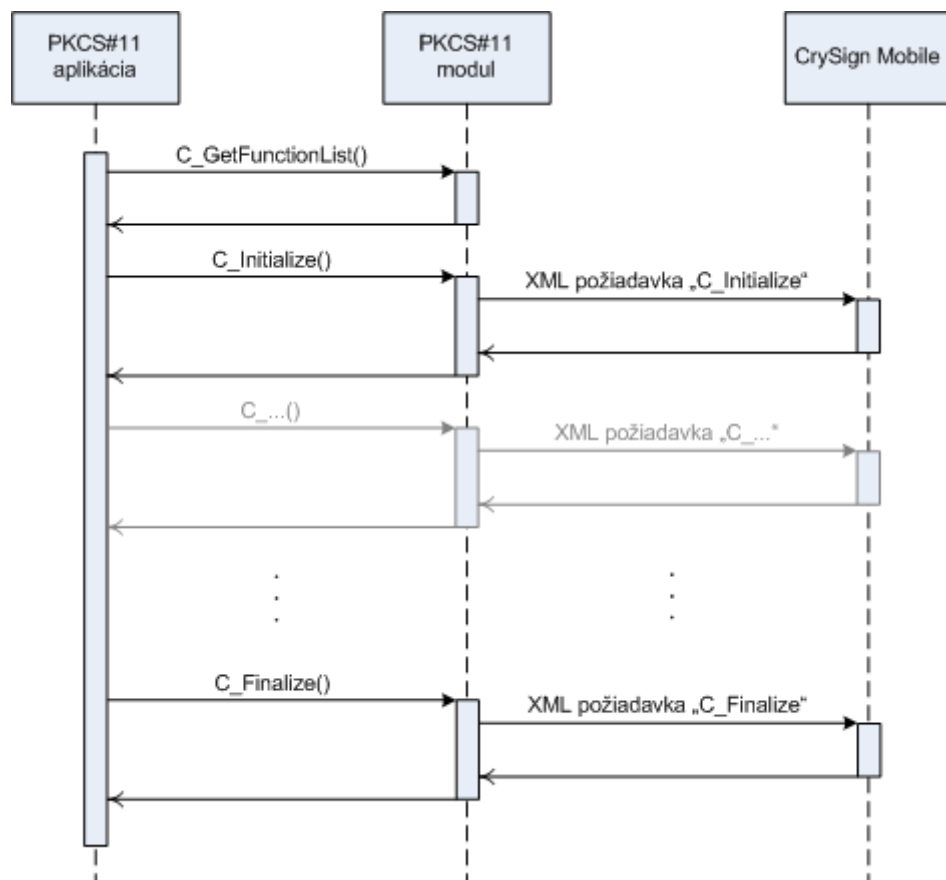
1. kontrola vstupných parametrov - napr. na hodnoty NULL_PTR a pod.,
2. vybudovanie XML požiadavky s jej parametrami,
3. odoslanie XML požiadavky,
4. prijatie XML odpovede,
5. spracovanie XML odpovede,
6. kontrola návratovej hodnoty funkcie z XML odpovede (ak nastalo zlyhanie funkcie, tak návrat tejto hodnoty PKCS#11 aplikácii),
7. prečítanie parametrov odpovede, ich umiestnenie do výstupných bufferov,
8. vrátenie príslušnej návratovej hodnoty (väčšinou CKR_OK).

Sekvenčný diagram znázorňujúci výmenu správ medzi časťami softvéru je znázorený na obrázku 5-1. Funkcie PKCS#11 modulu dostávajú od aplikácie už alokované buffery, do ktorých umiestňujú požadované dáta (výstupné parametre).



Obr. 5-1: Sekvenčný diagram PKCS#11 funkcií

Postupnosť volania PKCS#11 funkcií je znázornená v sekvenčnom diagrame na obrázku 5-2.



Obr. 5-2: Sekvenčný diagram postupnosti volania PKCS#11 funkcií

PKCS#11 aplikácia ako prvú zavolá funkciu *C_GetFunctionList*, ktorá vráti ukazovatele na všetky PKCS#11 funkcie. Týka sa to len PKCS#11 modulu, preto sa táto požiadavka do mobilného zariadenia neprenáša.

Pred použitím akejkoľvek inej funkcie (okrem *C_GetFunctionList*) je aplikácia povinná zavolať *C_Initialize*, čím dá modulu (a tým aj mobilnému zariadeniu) vedieť, že má záujem využívať jeho funkcie. V tejto chvíli sa na strane telefónu vytvorí inštancia PKCS#11 knižnice (*PKCS11Library*) prislúchajúca tejto aplikácii.

Po inicializácii môže aplikácia volať ostatné PKCS#11 funkcie. Môže si napríklad zistiť, aké má k dispozícii sloty (funkcia *C_GetSlotList*), otvoriť na jednom z nich reláciu (*C_OpenSession*), nájsť si kľúče a certifikáty (funkcie *C_FindObjects**), zistiť si o nich informácie (*C_GetAttributeValue*), použiť ich na digitálny podpis (*C_SignInit*, *C_Sign*) a ukončiť reláciu (*C_CloseSession*). V sekvenčnom diagrame (obr. 5-2) sú tieto funkcie zastúpené šedou farbou.

Keď aplikácia už nemá v úmysle používať PKCS#11 modul, dá mu to vedieť volaním funkcie *C_Finalize*. To spôsobí uvoľnenie všetkých prostriedkov asociovaných s touto aplikáciou ako na strane PKCS#11 modulu, tak i na strane mobilného telefónu - zaniknutie inštancie PKCS#11 knižnice priradenej tejto aplikácii.

5.2 NÁJDENIE MOBILNÉHO ZARIADENIA

Ako bolo spomenuté v kapitole 3.2 (o návrhu komunikácie), mobilné zariadenie v prípade pripojenia pomocou softvéru *Microsoft ActiveSync* má IP adresu 169.254.2.1 a počítač má adresu 169.254.2.2. PKCS#11 modul sa však nemôže spoľahnúť, že mobilný telefón bude vždy na tejto adrese. Napríklad ak sa vytvorí sieť nie pomocou *ActiveSync*, ale len pomocou Bluetooth PAN, IP adresy počítača a telefónu sú síce z rozsahu 169.254.0.0/16 ale sú iné. Preto PKCS#11 modul má zabudovaný mechanizmus zistenia mobilného zariadenia v sieti pomocou UDP broadcastov. To funguje nasledovne:

- Pri prvej požiadavke na komunikáciu s mobilným zariadením modul zistí, že ešte nepozná jeho IP adresu.
- Modul zistí pomocou volaní Windows API, aké všetky sieťové rozhrania sú v systéme dostupné, spolu s ich IP adresami a sieťovými maskami.
- Modul pre každé sieťové rozhranie zistí jeho broadcastovú adresu (logickým OR IP adresy rozhrania s invertovanou sieťovou maskou tohto rozhrania).

- Postupne na každú z týchto broadcastových adries odošle UDP datagram obsahujúci reťazec „CrySign_Mobile_Request“ a čaká 300 ms na odpoveď.
- Keď modul dostane odpoveď obsahujúcu reťazec „CrySign_Mobile_Response“, prestane prehľadávať a zapamätá si IP adresu, z ktorej túto odpoveď dostal. Každá ďalšia požiadavka na komunikáciu s mobilným zariadením bude prebiehať na túto IP adresu.
- Ak sa týmto spôsobom žiadne zariadenie nenašlo, modul vráti PKCS#11 programu návratovú hodnotu CKR_DEVICE_REMOVED (znamená, že zariadenie nie je pripojené k počítaču).

Keďže sa broadcasty odosielať na všetky sieťové rozhrania, mobilný telefón môže byť pripojený aj do rovnakej vnútornej IP siete, ako používa počítač trebárs na pripojenie k internetu, a stále ho PKCS#11 modul nájde. Toto je asi najčastejší prípad, keď počítač aj telefón sú pripojené cez Wi-Fi k tomu istému prístupovému bodu a majú IP adresy z rovnakej siete - tým pádom sa v tejto sieti doručujú broadcasty všetkým stanicam.

Čo však robiť, keď sa mobilné zariadenie nenachádza v rovnakej IP sieti, ako počítač? Odpoveď na túto otázku ponúka kapitola 5.3.

5.3 KONFIGURAČNÝ SÚBOR PKCS#11 MODULU

Pomocou konfiguračného súboru je možné nastaviť IP adresu, ktorá sa má použiť na komunikáciu s mobilným zariadením. Ak konfiguračný súbor existuje, nevykonáva sa prehľadávanie pomocou UDP broadcastov, ako je popísané v kapitole 5.2, ale komunikácia bude smerovať priamo na IP adresu špecifikovanú v tomto súbore. V prípade, že sa komunikácia na túto adresu nepodarí, modul vráti aplikácii návratovú hodnotu CKR_DEVICE_REMOVED.

Konfiguračný súbor sa nazýva *wmpkcs11.xml* a treba ho umiestniť do rovnakého adresára, v akom je spúšťačí súbor PKCS#11 aplikácie. Pre Mozilla Thunderbird by to znamenalo vytvorenie tohto súboru tam, kde je súbor *Thunderbird.exe*. Je to tak preto, že bežiaci PKCS#11 modul je počas použitia v podstate súčasťou aplikácie, ktorá ho využíva. Konfiguračný súbor je XML dokument v nasledujúcom tvare:

```
<?xml version="1.0" encoding="utf-8"?>
<config>
  <host>172.16.17.18</host>
</config>
```

Ako príklad je tu uvedená IP adresa zariadenia 172.16.17.18. Namiesto IP adresy je možné uviesť aj doménové meno hostiteľa. Týmto spôsobom je možné špecifikovať IP adresu mobilného telefónu aj v tom prípade, že by celá komunikácia mala prebiehať cez internet a cez množstvo smerovačov. Iba na konci cesty, pri mobilnom telefóne, je nutné zaistiť, aby bol dostupný „zvonka“ na portoch TCP a UDP číslo 57404 (nasmerovaním týchto portov ak je to potrebné).

Nastavenie IP adresy telefónu napevno v konfiguračnom súbore sa hodí aj vtedy, keď máme z počítača dostupných takýchto zariadení viac. Automatické vyhľadávanie by uprednostnilo ten telefón, ktorý pošle odpoveď na broadcast skôr, a to môže byť zakaždým iný telefón. Ďalšou výhodou využitia konfiguračného súboru je preskočenie vyhľadávania a tým čiastočné zrýchlenie štartu PKCS#11 aplikácie.

6 TESTOVANIE POČAS VÝVOJA

Softvérové vybavenie bolo treba počas implementácie testovať na reálnych zariadeniach. Práve testovaním na strane mobilného zariadenia i na strane PC sa zaoberá táto kapitola.

6.1 TESTOVANIE NA STRANE PC

Na strane PC je PKCS#11 modul v tvare DLL súboru, ktorý si aplikácia načíta do svojho pamäťového priestoru. Tento spôsob fungovania znemožňuje testovanie pomocou breakpointov (prerušení behu programu) vo vývojovom prostredí MS Visual Studio.

Čiastočnou náhradou a spôsobom, ako získať pomocné výpisy obsahu premenných, je zapisovať ich do pomocného súboru. Slúži na to funkcia *DebugIt* v súbore *pkcs11.cpp* a ako výstupný súbor predvolene používa *C:\wmpkcs11.txt* a jednotlivé záznamy sa pridávajú na jeho koniec. Spomenutá funkcia sa volá po vstupe a pred výstupom z každej PKCS#11 funkcie, preto ak v niektorej z funkcií nastane pád programu, z výstupného súboru možno vyčítať, v ktorej z nich sa to stalo. Volanie tejto funkcie nastáva aj v prípade, že niektorej z PKCS#11 funkcií boli posunuté nesprávne parametre. Funkcia *DebugIt* ale zapisuje do súboru len v prípade, že globálna premenná *Debug* je nastavená na TRUE (čo predvolene nie je).

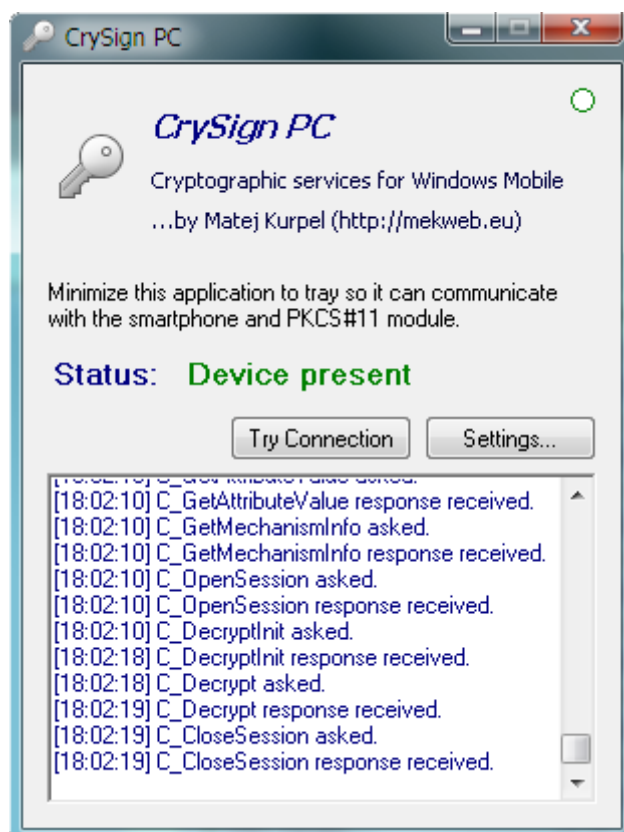
So zápisom do pomocného súboru by sa dalo vystačiť len ťažko; našťastie existuje PKCS#11 modul (*pkcs11-spy.dll*), ktorý umožňuje podrobne sledovať volanie funkcií vyvíjaného modulu, ich parametre a návratové hodnoty. Spomínaný modul je voľne dostupný v rámci softvéru *OpenSC*⁸. Jeho použitie je nasledujúce:

- v operačnom systéme nastavíme premenné prostredia (*environment variables*):
 - PKCS11SPY = (plná cesta k vyvíjanému PKCS#11 modulu)
 - PKCS11SPY_OUTPUT = (plná cesta k súboru pre zaznamenanie komunikácie)
- do PKCS#11 aplikácie načítame namiesto nami vyvíjaného modulu, modul *pkcs11-spy.dll*,
- spustíme PKCS#11 aplikáciu,
- ... a do súboru uvedeného v premennej prostredia PKCS11SPY_OUTPUT sa zaznamenávajú všetky volania PKCS#11 funkcií.

⁸ Domovská stránka projektu OpenSC: <http://www.opensc-project.org>

Spomínané riešenie má výhodu v jeho jednoduchosti a množstve informácií, ktoré sa do súboru zaznamenávajú. PKCS#11 hodnoty, ktoré sú priradené v hlavičkovom súbore, sú rovno prekladané späť na ich čitateľnejšiu podobu (napr. pri návratovej hodnote 0 sa uvedie CKR_OK). Tam, kde sa deje výmena binárnych dát, je rovno zobrazený výstup v hexadecimálnom formáte (a ak sa dá vypísať, tak je aj vypísaný v ASCII).

Posledným spôsobom testovania na strane PC bolo použitie softvéru vyvinutého špeciálne pre tento účel (s názvom *CrySign PC*). Tento program je implementovaný v C#.NET 3.5 a slúži ako „proxy“ medzi PKCS#11 modulom a mobilným zariadením. *CrySign PC* načúva na porte TCP 57404 a všetko, čo dostane, posunie ďalej mobilnému zariadeniu (a s odpoveďou, ktorú dostane, to funguje naopak). Tento program nedisponuje možnosťou automatického vyhľadania zariadení a preto treba IP adresu zariadenia zadať do jeho nastavení. Nastavením breakpointu na príslušnom mieste v *CrySign PC* sa dá sledovať XML dokument posielaný do mobilného zariadenia a rovnako aj odpoveď naň. Okrem toho aj program samotný sleduje túto komunikáciu a do pomocnej časti okna zaznamenáva dianie. Vzhľadom sa podobá na *CrySign Mobile* (obr. 6-1).



Obr. 6-1: Ukážka programu CrySign PC

Pre použitie *CrySign PC* treba nastaviť v konfiguračnom súbore PKCS#11 modulu adresu mobilného zariadenia na *localhost* (podľa kapitoly 5.3).

6.2 TESTOVANIE NA STRANE MOBILNÉHO ZARIADENIA

Na strane mobilného zariadenia bolo testovanie jednoduché, pretože vývojové prostredie Visual Studio 2008 poskytuje všetky potrebné funkcionality. Je možné si nastaviť breakpointy, sledovať obsah premenných, trasovať program a vykonávať mnoho iných činností, na ktoré je programátor zvyknutý pri vývoji aplikácii bežiacich na počítači.

Softvér *CrySign Mobile* bol testovaný na dvoch zariadeniach, a to:

- HTC Touch HD T8282 (OS Windows Mobile 6.1),
- HP iPAQ (OS Windows for PocketPC 2003).

Vývoj prebiehal iba na telefóne HTC Touch HD, a po jeho ukončení bol program odskúšaný na zariadení od HP (čo nie je ani telefón, iba PDA). Softvér *CrySign Mobile* fungoval aj na tomto zariadení bez problémov - to je práve požadovaný efekt implementácie tohto softvéru pod .NET Compact Frameworkom.

Digitálny podpis a dešifrovanie boli testované v e-mailovom klientovi Mozilla Thunderbird, generovanie RSA kľúčov a prihlásenie cez web pomocou osobného certifikátu boli testované v prehliadači Mozilla Firefox. Na testovanie prihlásenia pomocou certifikátu bol použitý webový portál „e-vzdelávanie“ Žilinskej univerzity⁹.

⁹ <https://vzdel.uniza.sk>

7 PODROBNOSTI VOLANIA PKCS#11 FUNKCIÍ

Táto kapitola popisuje, ktoré funkcie PKCS#11 modulu a s akými parametrami sa volajú pri rôznych činnostiach, ktoré užívateľ vykonáva. Použité záznamy volania funkcií sa kvôli ich veľkému rozsahu nachádzajú v prílohách. Tieto volania pochádzajú z výstupu modulu *pkcs11-spy.dll* (viac v kap. 6.1) a na ne sa odkazuje text pomocou čísel v zátvorkách. V softvéri *CrySign Mobile* bola načítaná trojica „certifikát + súkromný kľúč + verejný kľúč“ majúca spoločné ID (CKA_ID) = 1. Podkapitoly 7.1 až 7.3 sú spoločné pre Mozilla Thunderbird aj Mozilla Firefox (spoločne „PKCS#11 program“), podkapitoly 7.4 a 7.5 sa vzťahujú k Thunderbird-u, 7.6 a 7.7 k Firefox-u.

Na priloženom DVD sa nachádza programátorská príručka, ktorá obsahuje okrem volaní funkcií aj XML dokumenty tečúce medzi PC a mobilným zariadením.

7.1 SPUSTENIE PKCS#11 PROGRAMU

PKCS#11 volania prebiehajúce pri spustení aplikácie sú uvedené v prílohe 5. Pri spustení PKCS#11 programu tento musí zavolať ako prvú funkciu *C_GetFunctionList*, ktorá vráti smerníky na ďalšie funkcie PKCS#11 modulu (0).

Ďalším povinným volaním je *C_Initialize* (1, 2). V parametroch volania tejto funkcie je udaný spôsob, akým sa chystá aplikácia pristupovať k modulu z pohľadu používania vlákien a zamykania. Ak modul nedokáže v špecifikovanom režime fungovať, vráti CKR_CANT_LOCK.

Volanie funkcie *C_GetInfo* (3) vráti informácie o module. Aplikácia si pomocou *C_GetSlotList* zistí najprv počet slotov (4) a potom zoznam ich identifikátorov (5). K dispozícii je len jeden slot a ten má identifikátor 0.

Identifikátor slotu získaný volaním *C_GetSlotList* sa použil na zistenie informácií o danom slote volaním funkcie *C_GetSlotInfo* (6). Tieto informácie zahŕňajú výrobcu, popis, verziu a údaj o tom, že je to hardvérový slot a že v slote sa nachádza token. O tomto tokene si aplikácia zistí detaily volaním funkcie *C_GetTokenInfo* (7). Token má svojho výrobcu, model, popis, sériové číslo, verziu, množstvo pamäte (návratová hodnota -1 znamená „informácia nedostupná“), limit súčasne otvorených relácií (hodnota 0 znamená „bez limitu“), obmedzenie dĺžky PIN kódu (ktorý sa v tomto softvéri nepoužíva), verziu hardvéru i firmvéru, systémový čas a príznaky udávajúce, že autentifikácia prebieha

bezpečnou cestou (teda na zariadení a nie na PC), že token je už inicializovaný (nie je treba nastavovať PIN kód), a že v zariadení sú hodiny.

Rovnakým spôsobom, ako získala aplikácia zoznam slotov, teraz získa zoznam kryptografických mechanizmov volaním *C_GetMechanismList* (8, 9): najprv si zistí ich počet a potom ich identifikátory. Naše zariadenie má dva mechanizmy: jeden sa používa pre digitálny podpis a dešifrovanie, druhý na generovanie RSA kľúčov.

Aplikácia otvorí reláciu volaním *C_OpenSession* (10). Táto relácia je iba na čítanie, čo je určené parametrom *flags*. Vo výstupnom parametri *phSession* aplikácia dostane pridelené číslo novo otvorenej relácie, ktoré môže použiť v ďalších volaniach funkcií, ak sa potrebuje odkázať na túto reláciu.

Nasleduje vyhľadávanie objektov použitím relácie: najprv sa aplikácia pokúša nájsť objekty typu *CKO_NETSCAPE_BUILTIN_ROOT_LIST* volaním *C_FindObjectsInit* (11), ktoré však *CrySign Mobile* nepozná a tak vráti hodnotu *CKR_ATTRIBUTE_VALUE_INVALID*. Potom sa aplikácia pusti do vyhľadávania certifikátov uložených v tokene (12 - 14). To sa jej podarí a v zozname nájdených objektov je vrátený jeden certifikát, čo je objekt s identifikátorom 1.

Volaním funkcie *C_GetAttributeValue* (15, 16) aplikácia zistí jeho názov a tiež, že certifikát je uložený na tokene. Ďalšími volaniami tejto funkcie (17, 18) sa získajú ďalšie atribúty certifikátu: jeho typ, identifikátor, vydavateľ, subjekt, sériové číslo a tiež celý certifikát v binárnej (DER) podobe.

Zvyšné volania (19 - 23) sú pokusmi o získanie neštandardných atribútov a o nájdenie neštandardných objektov, ktoré *CrySign Mobile* nepozná a dáva to aplikácii najavo vrátením príslušnej návratovej hodnoty.

7.2 ZOBRAZENIE INFORMÁCIÍ O TOKENE A SLOTE

K informáciám o tokene a slote sa možno v programoch Mozilla Thunderbird a Mozilla Firefox dostať cez menu *Tools - Options... - Advanced - Certificates / Encryption - Security Devices*. Hneď pri otvorení okna s bezpečnostnými zariadeniami sa žiadne volania PKCS#11 funkcií nekonajú, pretože zobrazený názov tokenu a slotu sa zistil už pri spúšťaní aplikácie. Po kliknutí na názov zariadenia (t. j. názov slotu) v ľavej časti okna nastanú volania funkcií *C_GetSlotInfo* a *C_GetTokenInfo*, ktoré vyzerajú presne tak isto, ako volania 6 a 7 uvedené v prílohe 5 a popísané v predchádzajúcej kapitole.

7.3 ZOBRAZENIE ZOZNAMU CERTIFIKÁTOV

Zoznamy dostupných certifikátov sa v programoch Mozilla Thunderbird a Mozilla Firefox nachádzajú v menu *Tools - Options... - Advanced - Certificates / Encryption - View Certificates*. Osobné certifikáty ku ktorým užívateľ vlastní súkromný kľúč sa nachádzajú hneď v prvej záložke - *Your Certificates*, a práve tu sa zobrazí certifikát z mobilného telefónu. Volania funkcií, ktoré pri zobrazení zoznamu certifikátov nastanú, sú popísané v tejto kapitole a uvedené v prílohe 6.

Volaniami funkcií *C_FindObjects** (31-33) aplikácia nájde všetky certifikáty. V našom prípade je len jeden a ten má číslo 1. Ďalej pomocou funkcie *C_GetAttributeValue* (34 - 37) zistí atribúty tohto certifikátu, ako je jeho názov a ID (atribút *CKA_ID*, pomocou ktorého sa k sebe viažu certifikáty a kľúče). Aplikácia vyhľadá všetky súkromné kľúče s týmto ID (38 - 40). K dispozícii je jeden takýto súkromný kľúč a ten má číslo 2.

7.4 ODOSLANIE DIGITÁLNE PODPÍSANÉHO E-MAILU

Podrobnosti ohľadom volania jednotlivých PKCS#11 funkcií sú uvedené v prílohe 7. Pri odosielaní digitálne podpísaného e-mailu Mozilla Thunderbird najprv zistí, či k certifikátu máme aj súkromný kľúč (presne takým istým spôsobom, ako pri zobrazení zoznamu certifikátov - kapitola 7.3) (28 - 32). Pokúsi sa v zariadení nájsť zoznam neplatných certifikátov pre našu certifikačnú autoritu (33), ale zariadenie tento zoznam nepozná (je to neštandardný typ objektu - *CKA_CLASS = CKO_NETSCAPE_CRL*), preto je aplikácii vrátená hodnota *CKR_ATTRIBUTE_VALUE_INVALID*. Thunderbird ďalej nájde certifikát volaniami *C_FindObjects** (34 - 36) a to pomocou jeho binárnej podoby (hodnoty atribútu *CKA_VALUE*). V ďalších volaniach (37 - 41) opäť zistí existenciu súkromného kľúča k tomuto certifikátu a sériou volaní funkcie *C_GetAttributeValue* (42 - 47) zistí jeho vlastnosti: že je to RSA kľúč uložený na tokene, že je to súkromný objekt (jeho hodnota nesmie byť prezradená aplikácii), a tiež zistí jeho modulus kvôli tomu, aby sa dozvedel, aký veľký buffer má alokovať pre podpis (ten bude taký veľký, ako je modulus kľúča).

Pre podpísanie si Thunderbird otvorí druhú reláciu (48). Volaním *C_SignInit* (49) nad touto reláciou sa pripraví operácia podpisovania určením kľúča a algoritmu, ktorý sa má použiť. V tejto chvíli si *CrySign Mobile* vypýta od užívateľa heslo k súkromnému kľúču, aby ho bolo možné dešifrovať a použiť pre podpis. Ak nebolo od užívateľa získané správne heslo, táto funkcia vráti *CKR_OPERATION_CANCELED*, inak vráti *CKR_OK*.

Samotné podpísanie sa vykoná volaním funkcie *C_Sign* (50), kde sú poskytnuté dáta na podpísanie a buffer vyhradený pre podpis. Druhá relácia sa zatvorí (51) a Thunderbird sa pokúša ešte nájsť objekty s neznámymi atribútmi (52 - 54), avšak dostane oznámenie o neúspechu použitím návratovej hodnoty *CKR_ATTRIBUTE_VALUE_INVALID*.

7.5 ČÍTANIE ZAŠIFROVANÉHO E-MAILU

Mozilla Thunderbird sa pokúša dešifrovať e-mail už pri jeho prijatí, volania funkcií pri otváraní e-mailu užívateľom sú takmer rovnaké (tieto sú popísané v tejto kapitole a uvedené v prílohe 8).

Thunderbird najprv nájde certifikát podľa certifikačnej autority a sériového čísla (30 - 32). Získa niektoré jeho atribúty a nájde k nemu súkromný kľúč (33 - 39). Opäť nájde certifikát, tentoraz podľa jeho binárneho obsahu (40 - 42) a znova nájde súkromný kľúč (43 - 47). Volaniami *C_GetAttributeValue* (48 - 51) sa dozvie niektoré atribúty tohto kľúča.

Na dešifrovanie si Thunderbird otvorí novú reláciu (52). Volaním *C_DecryptInit* (53) sa nastaví kľúč a mechanizmus, ktorý sa použije na dešifrovanie. Táto funkcia zahŕňa vypýtanie si hesla k súkromnému kľúču od užívateľa. Ak nebolo poskytnuté správne heslo, návratová hodnota tejto funkcie je *CKR_OPERATION_CANCELED*, inak *CKR_OK* (rovnako ako pri podpisovaní). Samotné dešifrovanie sa vykoná volaním funkcie *C_Decrypt* (54) a potom sa zatvorí relácia, v ktorej sa dešifrovalo (55).

7.6 PRIHLÁSENIE POMOCOU OSOBNÉHO CERTIFIKÁTU NA WEBE

Na testovanie prihlásenia pomocou osobného certifikátu bola použitá webová stránka „e-vzdelávanie“ Žilinskej univerzity: <http://vzdel.uniza.sk>. Konkrétne volania PKCS#11 funkcií aplikáciou Mozilla Firefox sú uvedené v prílohe 9.

Po zvolení prihlásenia pomocou osobného certifikátu na spomínanej stránke Firefox vyhľadá všetky certifikáty nachádzajúce sa na tokene (23 - 25). V uvedenom príklade našiel jeden certifikát, takže získa niektoré jeho atribúty a nájde k nemu súkromný kľúč (26 - 37). V tomto okamihu zobrazí užívateľovi dialógové okno s výberom dostupných certifikátov (teraz bude obsahovať len jeden certifikát) aj s ich podrobnosťami. Po potvrdení výberu certifikátu pokračuje Firefox vo volaní PKCS#11 funkcií; nájde certifikát podľa jeho binárnej podoby a k nemu súkromný kľúč (38 - 45). O tomto kľúči si zistí informácie - že je to súkromný objekt na tokene, že je to RSA kľúč a tiež zistí jeho

modulus (46 - 51) kvôli zisteniu veľkosti buffera potrebného na podpis (prihlásenie pomocou certifikátu je v tomto prípade podpis hashu zloženého z hashov SHA-1 a MD5). Otvorí sa nová relácia (52), inicializuje sa podpisovanie pomocou daného súkromného kľúča a s použitím mechanizmu CKM_RSA_PKCS (53), a od užívateľa sa získa heslo k súkromnému kľúču. Nasleduje volanie funkcie *C_Sign* (54), ktorá poskytne digitálny podpis Firefox-u a celá operácia sa ukončí zatvorením relácie, v ktorej sa podpisovanie vykonávalo (55). Užívateľ je v tejto chvíli úspešne prihlásený na webovú stránku.

7.7 GENEROVANIE RSA KĽÚČOV

Generovanie RSA kľúčov vyvoláva prehliadač Mozilla Firefox pri požiadavke na vygenerovanie kľúčov, keď užívateľ odosiela žiadosť o osobný certifikát na stránke certifikačnej autority. Je však možné si generovanie RSA kľúčov vyskúšať bez odosielania verejného kľúča niektorej certifikačnej autorite - spôsob je popísaný v kapitole 8.6. Táto kapitola popisuje len volanie PKCS#11 funkcií počas generovania kľúčov. Podrobnosti volaní sú uvedené v prílohe 10.

Na začiatku si Firefox vyžiada informácie o mechanizme CKM_RSA_PKCS na slot 0 - na to sa použije funkcia *C_GetMechanismInfo* (12). Na generovanie sa otvorí nová relácia (13), ktorá je určená aj na zápis (pomocou *flags*). Samotné generovanie sa spustí volaním funkcie *C_GenerateKeyPair* (14). Ako mechanizmus sa použije druhý podporovaný - CKM_RSA_PKCS_KEY_PAIR_GEN. Okrem požiadaviek na atribúty nových kľúčov je vstupným parametrom do tejto funkcie dĺžka kľúča v atribúte CKA_MODULUS_BITS - v našom prípade je to 1024 bitov. Verejný exponent bol určený atribútom CKA_PUBLIC_EXPONENT a jeho hodnota je 65537. Výstupnými parametrami sú identifikátory PKCS#11 objektov, ktoré predstavujú novo vygenerované kľúče.

Nasledujú volania *C_GetAttributeValue*, ktorými Firefox zisťuje atribúty verejného kľúča (15 - 18) a volania *C_SetAttributeValue*, ktoré majú zaistiť pridelenie rovnakého atribútu CKA_ID obom kľúčom (19, 20). Nakoniec sa zatvorí relácia použitá na generovanie kľúčov (21) a pokračuje sa opakovaným volaním *C_GetAttributeValue* kvôli zisteniu niektorých vlastností čerstvo vygenerovaného súkromného kľúča (22 - 25).

Keď sa odosiela žiadosť o certifikát certifikačnej autorite, celá táto žiadosť sa podpíše súkromným kľúčom a preto nasleduje klasický proces digitálneho podpisovania:

otvorí sa nová relácia (26), inicializuje sa podpisovanie novo vygenerovaným kľúčom a algoritmom RSA (27), vykoná sa samotné podpisovanie (28) a relácia sa zatvorí (29).

Užívateľ je žiadaný o zadanie hesla k súkromnému kľúču raz pri jeho generovaní (keď je kľúč týmto heslom zašifrovaný), a druhý krát v procese digitálneho podpisovania žiadosti o certifikát.

8 OVERENIE RIEŠENIA

Táto kapitola popisuje, ako nainštalovať a odskúšať softvér naprogramovaný v rámci tejto diplomovej práce. Všetky súbory potrebné na inštaláciu sa nachádzajú na priloženom DVD v adresári *bin*. Uživateľská príručka v anglickom jazyku je v súbore *CrySign.chm*¹⁰. Zdrojové kódy sa nachádzajú v adresári *src*.

Požiadavky softvéru sú nasledujúce:

- mobilné zariadenie s operačným systémom rodiny Windows,
- .NET Compact Framework nainštalovaný na mobilnom zariadení,
- program ovládajúci PKCS#11 (Mozilla Thunderbird, Mozilla Firefox),
- knižnice *Visual C++ 2008 SP1 Redistributable* nainštalované na počítači, kde sa bude používať PKCS#11 modul (priložené na DVD v adresári *Redist*),
- existujúce TCP/IP spojenie medzi počítačom a mobilným zariadením.

Inštalácia softvéru by mala prebehnúť v tomto poradí:

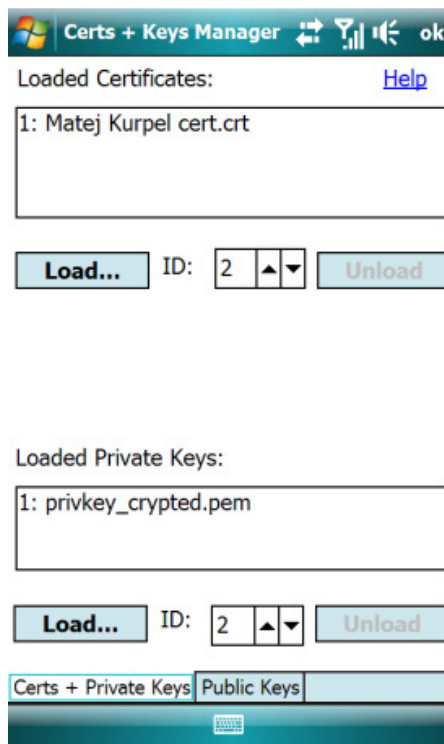
1. inštalácia a nastavenie softvéru CrySign Mobile do telefónu,
2. inštalácia PKCS#11 modulu do aplikácie na strane PC (tu Mozilla Thunderbird),
3. nastavenie aplikácie na strane PC.

8.1 INŠTALÁCIA A NASTAVENIE SOFTVÉRU NA STRANE TELEFÓNU

Inštalácia softvéru *CrySign Mobile* spočíva v prenesení inštalačného súboru *CrySign Mobile.cab* do telefónu a jeho spustení. Inštalátor extrahuje súbory na potrebné miesto (*Program Files\CrySign Mobile*) a umiestni zástupcu spustiteľného súboru do ponuky *Štart menu -> Programy*.

Pred nastavením softvéru *CrySign Mobile* treba do telefónu nakopírovať súkromný kľúč a k nemu prislúchajúci certifikát a potom ich nastaviť v menu *Certs + Keys Manager*. Je dôležité prideliť im rovnaké ID, aby program vedel, že patria k sebe. Formáty kľúčov a certifikátov podporované v softvéri *CrySign Mobile* sú uvedené v kapitole 4.2. Ukážka, ako by mal vyzeráť formulár s nastaveným súkromným kľúčom a certifikátom, je na obrázku 8-1.

¹⁰ CHM súbor bol vytvorený aplikáciou *Help and Manual*, ktorej inštalačný súbor sa nachádza na priloženom DVD. Tento program je voľne dostupný a plne funkčný po dobu 30 dní od inštalácie.



Obr. 8-1: Certifikát a súkromný kľúč k nemu prislúchajúci v CrySign Mobile

Dôležité je zopakovať, že ak súkromný kľúč v PEM formáte nebol šifrovaný, aplikácia požiada užívateľa o heslo a toto heslo použije na jeho zašifrovanie. Pôvodný nešifrovaný kľúč bude v pamäti telefónu nahradený jeho šifrovanou verziou. Ak už pri načítavaní bol zašifrovaný a jeho formát je podporovaný (viď kap. 4.2), užívateľ je požiadaný o heslo kvôli načítaniu verejného modulusu.

V správcovi certifikátov a kľúčov je možné na druhej záložke nastaviť aj verejný kľúč, ten sa ale v žiadnej z podporovaných kryptografických funkcií nepoužíva a táto možnosť je tu len kvôli generovaniu RSA kľúčov, kedy vygenerované kľúče sú rovno nastavené a pripravené na použitie v softvéri *CrySign Mobile*.

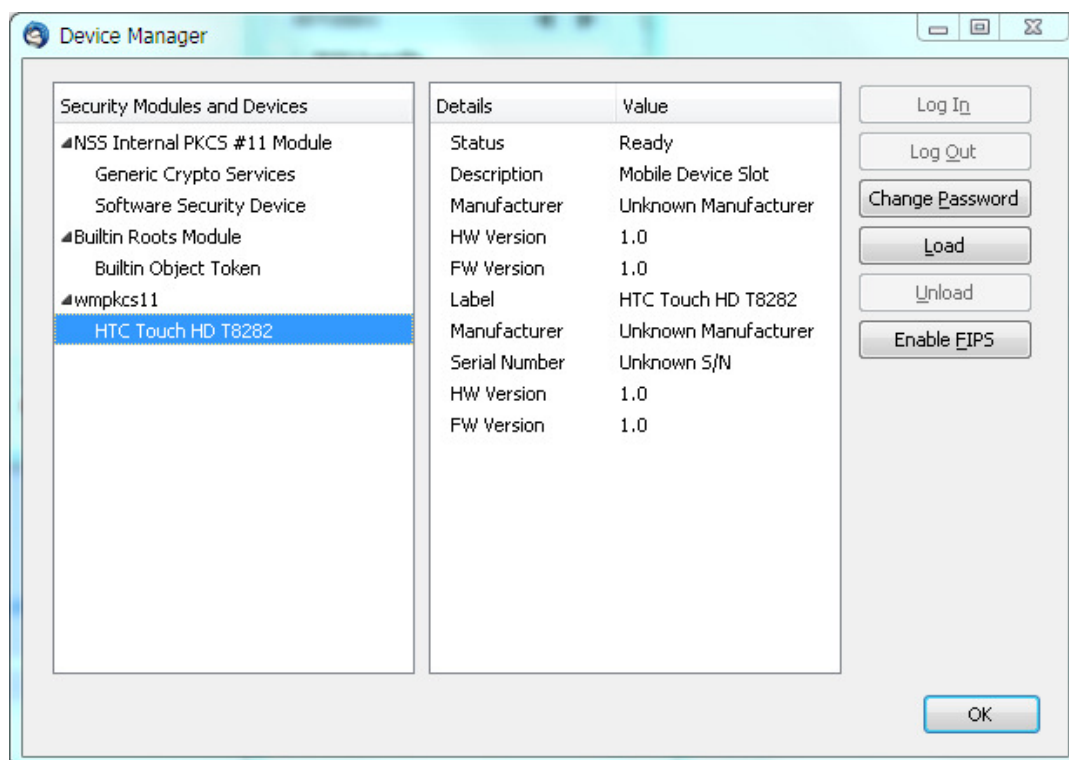
Po nastavení kľúčov a certifikátov treba aplikáciu *CrySign Mobile* reštartovať a od tej chvíle je pripravená na použitie. Ďalším krokom je nastavenie PKCS#11 aplikácie na strane PC.

8.2 NASTAVENIE E-MAIL. KLIENTA MOZILLA THUNDERBIRD

Pre účely tohto návodu budeme predpokladať, že PKCS#11 aplikácia je e-mailový klient Mozilla Thunderbird (v anglickom jazyku a vo verzii 3.1.9, ktorá bola aktuálna v čase písania tejto práce). Inštalácia pozostáva z načítania PKCS#11 modulu do tejto aplikácie a z nastavenia certifikátov na podpis (vrátane inštalácie certifikátu certifikačnej autority do zoznamu dôveryhodných certifikačných autorít).

8.2.1 Inštalácia PKCS#11 modulu

Zoznam aktuálne načítaných PKCS#11 modulov sa zobrazí postupom cez menu *Tools -> Options -> Advanced -> Certificates -> Security Devices*. Tu sú už nejaké vstavané moduly predinštalované, ten náš nainštalujeme kliknutím na tlačidlo *Load*. Ako názov modulu možno zvoliť čokoľvek, čo neobsahuje diakritiku ani špeciálne znaky (to by spôsobilo problémy kvôli známej chybe v Thunderbird-e). Modul nájdeme kliknutím na tlačidlo *Browse*. Po stlačení tlačidla *OK* začnú prvé volania PKCS#11 funkcií a začne prebiehať komunikácia s mobilným telefónom. Ak telefón nie je dostupný, alebo na ňom nebeží softvér *CrySign Mobile*, modul sa síce nainštaluje, ale nebude použiteľný pre nami požadované účely. Obrázok 8-2 ukazuje správne načítaný PKCS#11 modul.

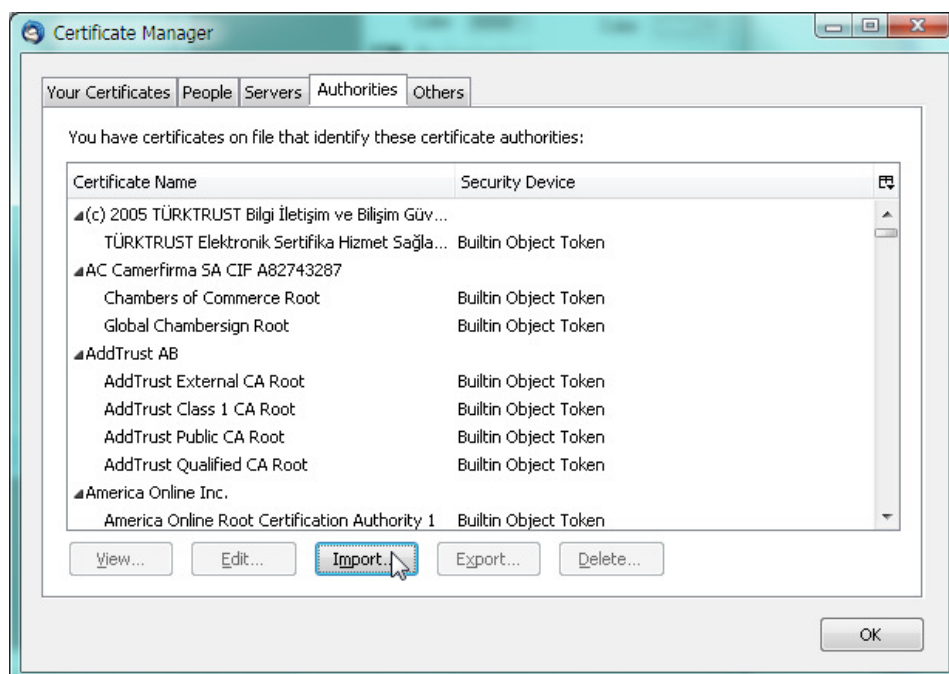


Obr. 8-2: PKCS#11 modul načítaný v aplikácii Mozilla Thunderbird

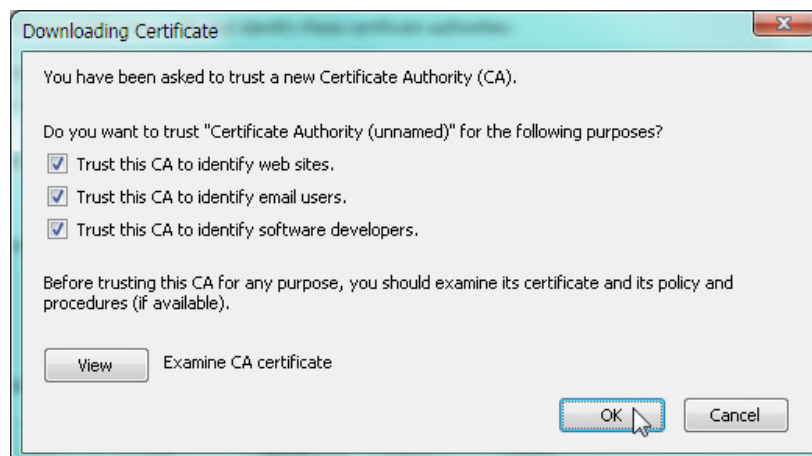
8.2.2 Inštalácia certifikátu certifikačnej autority

Aby bolo možné používať náš osobný certifikát na digitálne podpisovanie e-mailov, treba nastaviť dôveru certifikačnej autorite, ktorá náš certifikát vydala.

V Thunderbird-e je to možné urobiť pomocou správcu certifikátov, dá sa k nemu dostať cez menu *Tools -> Options -> Advanced -> Certificates -> View Certificates*. Dôveryhodné certifikačné autority sú uvedené v záložke *Authorities* (obr. 8-3). Tu treba kliknúť na tlačidlo *Import* a nájsť certifikát certifikačnej autority (ten treba najprv stiahnuť z jej webovej stránky alebo získať inak). Po otvorení certifikátu sa zobrazí okno, v ktorom treba zaškrtnúť políčka a povoliť tak dôveru tejto autorite (obr. 8-4). Po stlačení *OK* by mala byť táto certifikačná autorita pridaná do zoznamu dôveryhodných CA.



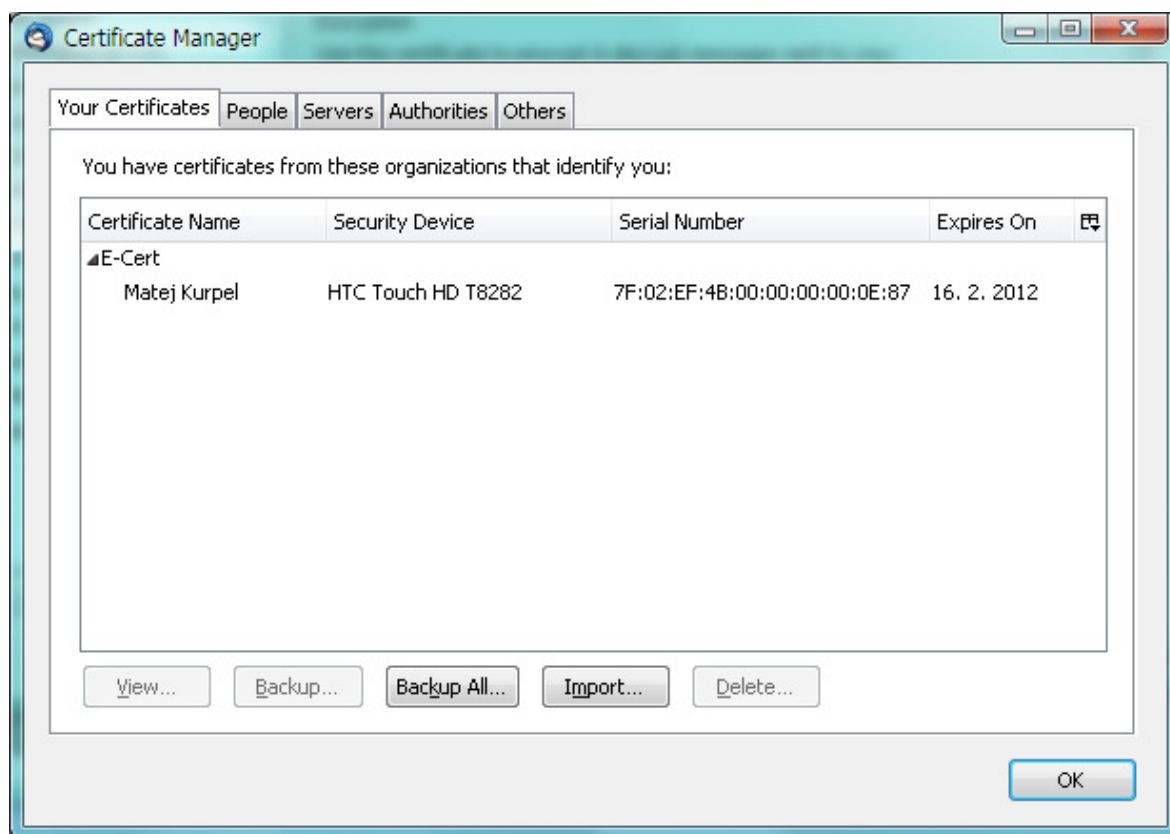
Obr. 8-3: Zoznam dôveryhodných certifikačných autorít v Mozilla Thunderbird



Obr. 8-4: Dialógové okno na vyslovenie dôvery novej certifikačnej autorite v Mozilla Thunderbird

8.2.3 Predvolenie osobného certifikátu pre šifrovanie a digitálny podpis

Osobný certifikát, ktorý sa nachádza v telefóne, by mal byť viditeľný v správcovi certifikátov: *Tools -> Options -> Advanced -> Certificates -> View Certificates* (záložka *Your Certificates* - obr. 8-5).

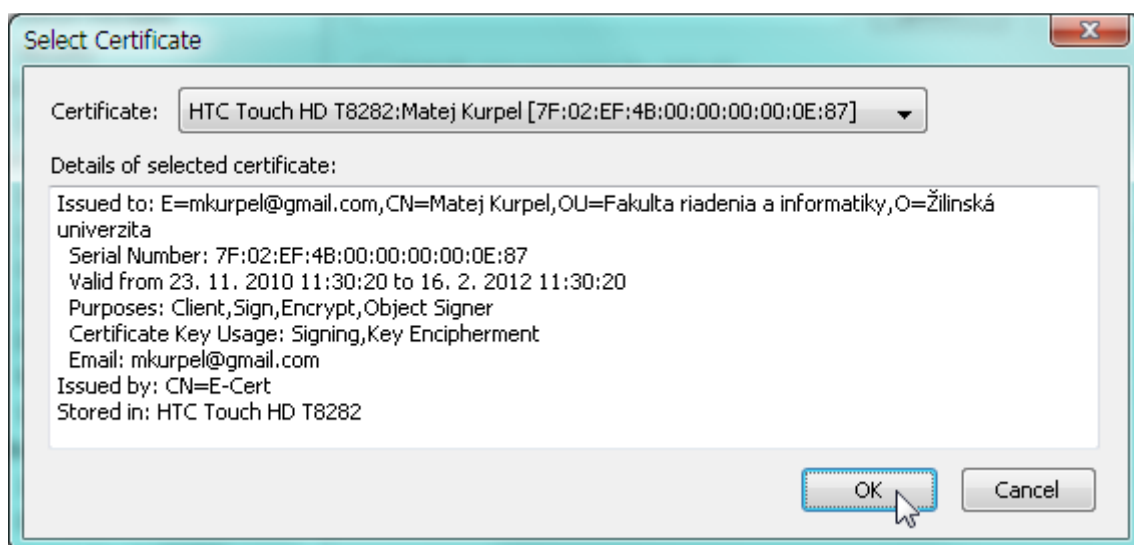


Obr. 8-5: Osobný certifikát z telefónu zobrazený v Mozilla Thunderbird

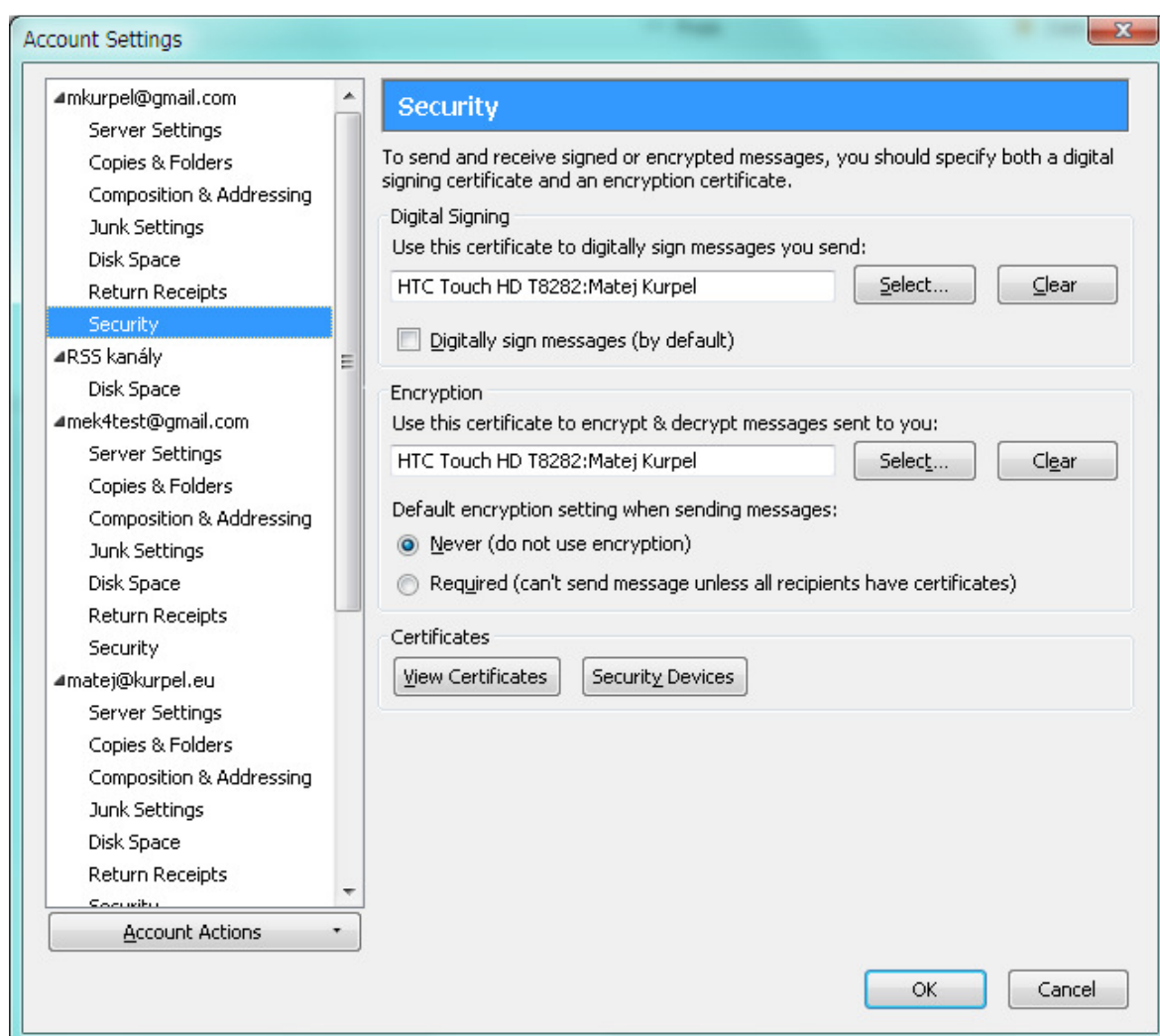
Podrobnosti o tomto certifikáte je možné zobrazíť dvojklikom naň.. Ak je certifikačná autorita, ktorá ho vydala (na obrázku E-Cert) dôveryhodná, tak v detailoch o certifikáte by malo byť napísané, že bol overený pre uvedené použitie.

Nastavenie tohto certifikátu pre digitálny podpis a šifrovanie je možné cez dialógové okno nastavenia e-mailového účtu, v ktorom chceme tento certifikát používať. E-mailová adresa tohto účtu by sa mala zhodovať s e-mailovou adresou uvedenou v osobnom certifikáte. Do tohto dialógového okna sa vstupuje pomocou menu *Tools -> Accounts -> Security*. V tomto okne treba kliknúť na tlačidlo *Select* pri voľbe *Digital Signing*. Nasleduje výber z dostupných certifikátov (obr. 8-6), z ktorého si vyberieme certifikát a potvrdíme tlačidlom *OK*. Na dodatočnú otázku, či chceme rovnaký certifikát použiť na šifrovanie, odpovieme, že áno.

Ukážka okna *Security* s nastaveným certifikátom je na obrázku 8-7.



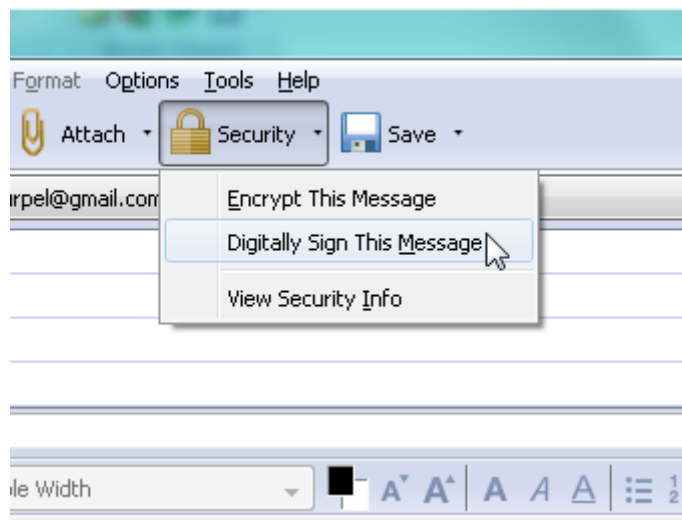
Obr. 8-6: Okno s výberom osobného certifikátu pre digitálny podpis



Obr. 8-7: Okno s nastavením osobného certifikátu pre šifrovanie a digitálny podpis

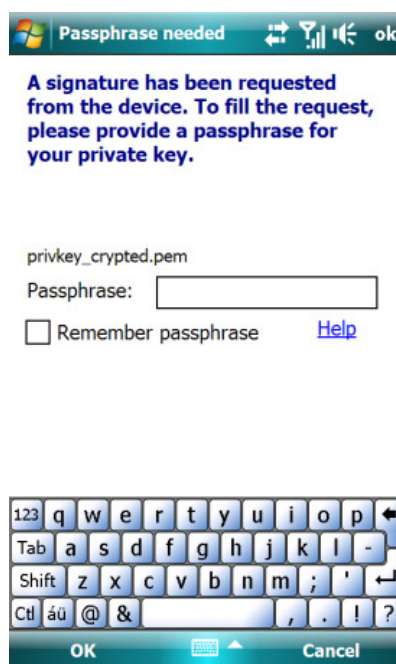
8.3 ODOSLANIE A PRIJATIE DIGITÁLNE PODPÍSANÉHO E-MAILU

Na odoslanie podpísaného e-mailu stačí v programe Mozilla Thunderbird zvoliť, že si želáme aktuálne vytváraný e-mail podpísať. Robí sa to pomocou menu *Security* -> *Digitally Sign This Message* (obr. 8-8).



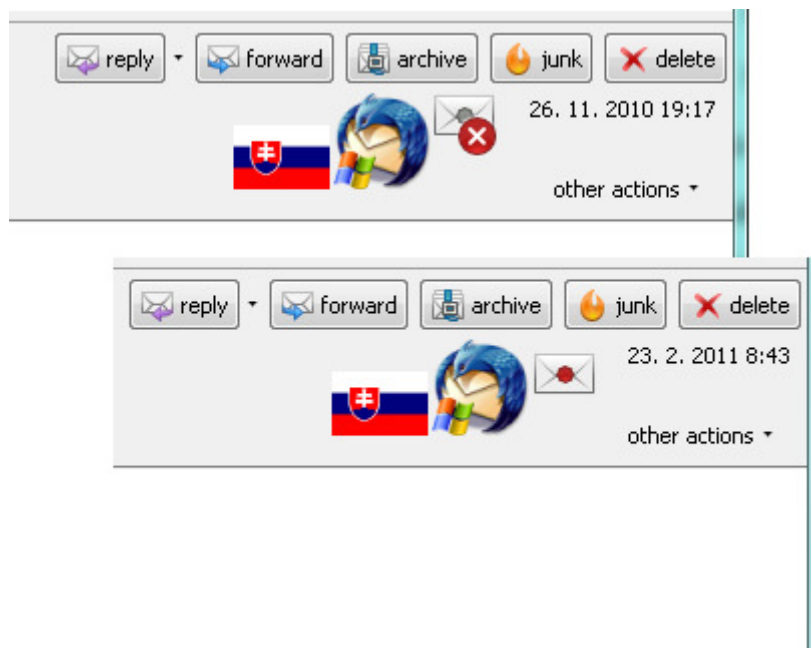
Obr. 8-8: Voľba podpísania e-mailu v programe Mozilla Thunderbird

Po zvolení tejto voľby sa v pravom dolnom rohu okna objaví ikona zapečatenej obálky. Potom už stačí mail len odoslať, ako zvyčajne. Po chvíli si softvér *CrySign Mobile* vypýta heslo k súkromnému kľúču, aby ho mohol použiť na podpísanie (obr. 8-9). Len čo užívateľ vloží heslo na displeji mobilného telefónu, e-mail sa podpíše a odošle.



Obr. 8-9: Požiadavka softvéru CrySign Mobile na heslo k súkromnému kľúču pri podpisovaní

Prijatie digitálne podpísaného e-mailu sa nelíši od prijatia nepodpísaného e-mailu, avšak Mozilla Thunderbird užívateľovi zobrazí, či je podpis platný. Tento fakt je indikovaný ikonou obálky, po kliknutí na ktorú je možné zobraziť podrobnosti a preskúmať osobný certifikát, ktorý bol k tejto správe pri podpisovaní priložený. Rozdiel medzi zobrazením platného a neplatného podpisu je jasný z obrázka 8-10. Po kliknutí na obálku reprezentujúcu platný digitálny podpis sa otvorí okno s podrobnosťami podobné tomu na obrázku 8-11.



Obr. 8-10: Neplatný a platný digitálny podpis v programe Mozilla Thunderbird



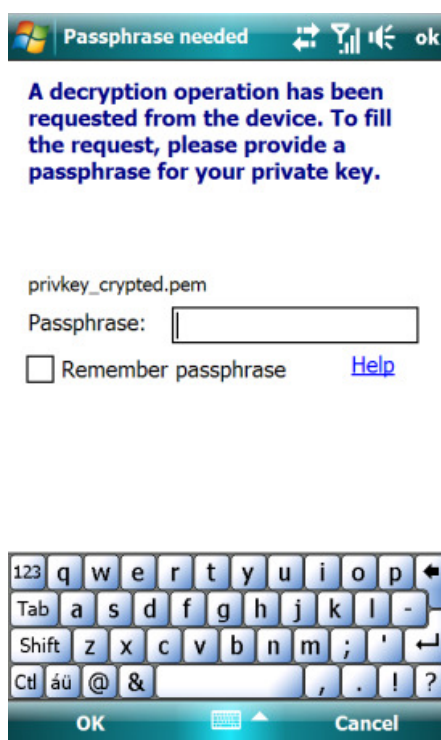
Obr. 8-11: Podrobnosti platného digitálneho podpisu

8.4 ODOSLANIE A PRIJATIE ZAŠIFROVANÉHO E-MAILU

Ak chceme v e-mailovom klientovi Mozilla Thunderbird odoslať šifrovaný e-mail, zvolíme to pri jeho vytváraní pomocou menu *Security -> Encrypt This Message* (ako pri digitálnom podpise na obr. 8-8). Po tomto sa v pravom dolnom rohu objaví ikona zámku, po kliknutí na ktorú sa zobrazí okno s informáciami o zabezpečení - z toho je dôležité najmä, či máme v Thunderbird-e nainštalovaný osobný certifikát príjemcu, pretože bez neho mu nemožno odoslať šifrovaný e-mail. Ak je všetko v poriadku, e-mail sa bez problémov odošle ako zvyčajne. K tomuto nie je treba súkromný kľúč odosielateľa a preto sa ani kľúč z mobilného telefónu nepoužije.

Súkromný kľúč naopak treba pri prijatí zašifrovaného e-mailu. Keďže ten bol zašifrovaný našim verejným kľúčom (získaným z nášho osobného certifikátu), obsah e-mailu môžeme získať len my pomocou nášho súkromného kľúča.

Pri prijatí takéhoto šifrovaného e-mailu sa Thunderbird pokúsi ho dešifrovať a to použitím mobilného telefónu. Softvér *CrySign Mobile* si vypýta heslo k súkromnému kľúču (obr. 8-12) a keď ho užívateľ zadá na telefóne, dešifrovaný e-mail sa zobrazí v počítači. Že e-mail bol zašifrovaný, pripomína ikonka zámku, po kliknutí na ktorú možno zobraziť podrobnosti.

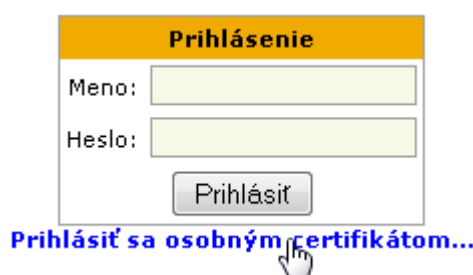


Obr. 8-12: Požiadavka softvéru CrySign Mobile na heslo k súkromnému kľúču pri dešifrovaní

8.5 PRIHLÁSENIE POMOCOU OSOBNÉHO CERTIFIKÁTU NA WEBE

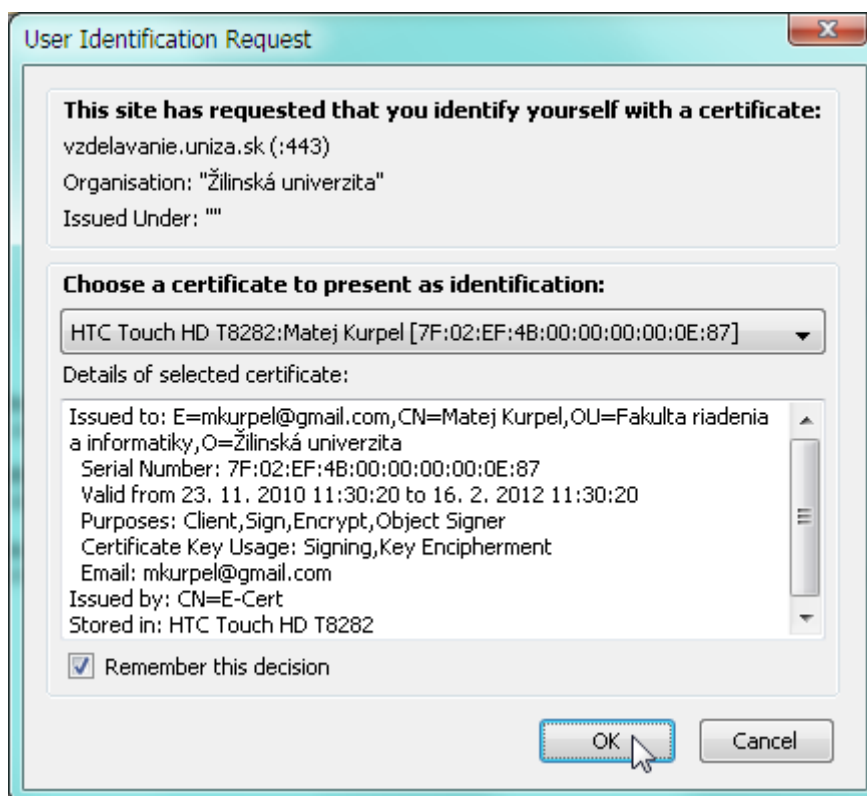
Pomocou súkromného kľúča a certifikátu v mobilnom telefóne sa dá aj prihlásiť na webovú stránku, podporujúcu takúto formu prihlásenia. Na testovanie bola použitá stránka „e-vzdelávanie“ Žilinskej univerzity: <http://vzdel.uniza.sk>. Samozrejme, ešte predtým treba v prehliadači Mozilla Firefox načítať PKCS#11 modul. Postup je rovnaký, ako v programe Mozilla Thunderbird - návod v kapitole 8.2.1.

Na webovej stránke zvolíme možnosť prihlásenia pomocou certifikátu (obr. 8-13).



Obr. 8-13: Prihlasovací formulár webového portálu „e-vzdelávanie“

Po krátkej komunikácii s mobilným telefónom prehliadač nájde osobný certifikát (alebo viac certifikátov) a zobrazí dialógové okno s výberom certifikátov pre autentifikáciu voči tejto webovej stránke (obr. 8-14).

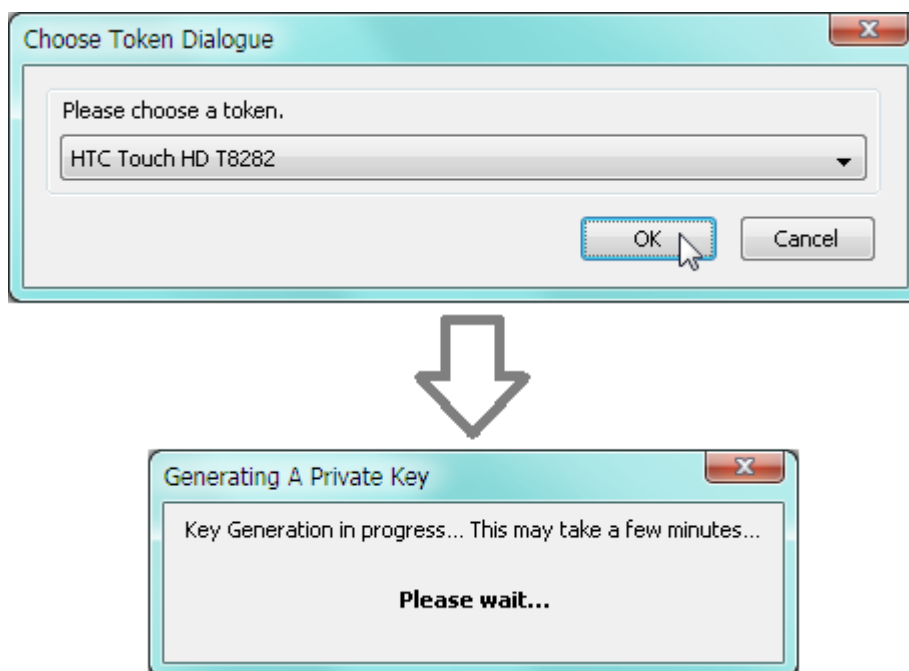


Obr. 8-14: Okno s výberom certifikátu pre autentifikáciu na webovej stránke

Po potvrdení je užívateľ požiadaný o heslo k súkromnému kľúču patriacemu k tomuto certifikátu a ak bolo zadané heslo správne, užívateľ je na stránku úspešne prihlásený.

8.6 GENEROVANIE RSA KLÚČOV

RSA kľúče sa zvyčajne generujú pri odosielaní žiadosti o certifikát na stránkach certifikačnej authority. V tejto žiadosti sa spolu s údajmi o osobe posiela čerstvo vygenerovaný verejný kľúč a všetko je podpísané príslušným súkromným kľúčom. Zo strany užívateľa prebieha generovanie kľúčov tak, že najprv je ponúknutý na výber zoznam zariadení podporujúcich generovanie kľúčov, z ktorých si užívateľ vyberie (obr. 8-15). Po odsúhlasení nastane samotné generovanie kľúčov a potom odoslanie formulára certifikačnej autorite. V softvéri *CrySign Mobile* na strane telefónu je užívateľ v tejto chvíli požiadaný o heslo na zašifrovanie vygenerovaného súkromného kľúča. Vygenerované kľúče sa ukladajú do adresára *GeneratedKeyPairs* v adresári s nainštalovaným programom *CrySign Mobile* v telefóne.



Obr. 8-15: Priebeh generovania RSA kľúčov v prehliadači Mozilla Firefox

Je možné vyskúšať si generovanie kľúčov bez použitia web stránky niektorej certifikačnej authority načítaním vlastnoručne vytvoreného PHP skriptu (alebo HTML stránky, ak netreba vidieť dáta prijaté serverom z formulára). HTML kód tejto stránky obsahuje formulár so špeciálnym elementom *keygen*, ktorý je pri zobrazení stránky

reprezentovaný políčkom na výber prehliadačom podporovaných dĺžok kľúčov. Tento element je bližšie popísaný na stránkach Mozilla Development Center (MDC)¹¹.

PHP súbor na vyskúšanie generovania kľúčov (pomenovaný *keygen.php*) môže vyzeráť napríklad takto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>KEYGEN TEST</title>
</head>
<body>
  <pre>
  <?php
  print_r($_POST);
  ?>
  </pre>
  <form action="keygen.php" method="post">
    <keygen name="key" challenge="ChallengeString74" />
    <input type="submit" />
  </form>
</body>
</html>
```

Po nahratí tohto súboru na server a jeho načítaní v prehliadači Mozilla Firefox by mala výsledná stránka vyzeráť podobne, ako ukazuje obrázok 8-16.



```
Array
{
}

High Grade Submit Query
```

Obr. 8-16: Testovacia web stránka na generovanie RSA kľúčov

Text nad výberom dĺžky kľúčov predstavuje výpis údajov odoslaných pomocou formulára - momentálne nie sú žiadne údaje k dispozícii, pretože formulár nebol odoslaný. Kliknutím na tlačidlo sa formulár odošle a začne sa generovanie kľúčov. Po ich úspešnom vygenerovaní a odoslaní formulára sa v texte nad formulárom zobrazí reťazec zakódovaný v Base64, ktorý bol odoslaný na server. Sú to údaje z formulára spolu s verejným kľúčom a podpisom, a to všetko kódované v DER (kapitola 2.5.4).

¹¹ MDC o HTML elemente keygen: <https://developer.mozilla.org/En/HTML/Element/keygen>

9 ZÁVER

Cieľom tejto diplomovej práce bolo navrhnuť a implementovať programové vybavenie pre PC a mobilný telefón s platformou Windows Mobile, poskytujúce služby pre digitálny podpis a šifrovanie. Riešenie malo spĺňať štandard PKCS#11.

Výsledný softvér sa skladá z PKCS#11 modulu na strane PC a z programu na strane mobilného zariadenia (*CrySign Mobile*). Toto riešenie umožňuje okrem spomenutých dvoch funkcií aj prihlásenie na webové stránky pomocou osobného certifikátu a generovanie RSA kľúčov, a tiež zlepšuje užívateľské pohodlie automatickým vyhľadáváním mobilného zariadenia. Implementácia nebola jednoduchá, bolo treba naštudovať špecifikáciu PKCS#11 a ďalšie štandardy. Zdrojové kódy softvéru sa nachádzajú na priloženom DVD v adresári *src* a skompilovaný softvér na distribúciu spolu s užívateľským manuálom v anglickom jazyku je uložený v adresári *bin*.

Softvér bol testovaný na dvoch zariadeniach (HTC Touch HD T8282 a HP iPAQ h2200) a bol funkčný na oboch bez zmien, čo je výhoda jeho implementácie pod .NET Compact Frameworkom.

Ďalší rozvoj tohto softvéru by mohol spočívať v šifrovaní komunikácie medzi PC a mobilným zariadením, keďže táto komunikácia môže prebiehať aj bezdrôtovo, či po verejných sieťach ako je internet. Tiež by bolo vhodné testovať jeho funkcie na ďalších programoch, ktoré ovládajú štandard PKCS#11 a odhaľovať prípadné nedostatky vo funkcionalite či implementovať nové PKCS#11 funkcie, ktoré by tieto programy potrebovali volať. Jednou z budúcich úloh by mohla byť úprava PKCS#11 modulu tak, aby sa okrem OS Windows dal použiť aj na Mac/Unix/Linux-ových systémoch.

V implementácii mobilného softvéru pre ďalšie platformy (Android, Symbian) budú pokračovať študenti inžinierskeho štúdia v rámci projektovej výučby na Fakulte riadenia a informatiky Žilinskej univerzity. Pre týchto študentov sa na priloženom DVD nachádza programátorská príručka.

10 ZOZNAMY

10.1 ZOZNAM POUŽITÝCH SKRATIEK

API	- Application Programming Interface
ASCII	- American Standard Code for Information Interchange
ASN.1	- Abstract Syntax Notation One
BER	- Basic Encoding Rules
CA	- Certifikačná autorita
CBC	- Cipher Block Chaining
CRL	- Certificate Revocation List
CSP	- Cryptographic Service Provider
DER	- Distinguished Encoding Rules
DES	- Data Encryption Standard
DLL	- Dynamic Link Library
DVD	- Digital Versatile Disc
ID	- Identifikátor
MD	- Message Digest
MIME	- Multipurpose Internet Mail Extensions
MSDN	- Microsoft Developer Network
NSS	- Network Security Services
OCSP	- Online Certificate Status Protocol
PAN	- Personal Area Network
PDA	- Personal Data Assistant
PEM	- Privacy Enhanced Mail
PIN	- Personal Identification Number
PKCS	- Public Key Cryptography Standards
PKI	- Public Key Infrastructure
TCP/IP	- Transmission Control Protocol / Internet Protocol
UDP	- User Datagram Protocol
USB	- Universal Serial Bus
UTF-8	- Unicode Transformation Format-8
VS2008	- Visual Studio 2008
XML	- Extensible Markup Language

10.2 ZOZNAM POUŽITÝCH ZDROJOV

- [1] Kryptológia. *Wikipedia, the free encyclopedia*. [Online] 23. január 2011.
<http://sk.wikipedia.org/wiki/Kryptol%C3%B3gia>
- [2] Kerckhoff's Principle. *Wikipedia, the free encyclopedia*. [Online] 7. február 2011.
http://en.wikipedia.org/wiki/Kerckhoffs%27s_Principle
- [3] Elektronický podpis. *Wikipedia, the free encyclopedia*. [Online] 9. január 2011.
http://cs.wikipedia.org/wiki/Elektronick%C3%BD_podpis
- [4] *PKI*. *Wikipedia, the free encyclopedia*. [Online] 24. december 2010.
<http://cs.wikipedia.org/wiki/PKI>
- [5] RFC 5751 (Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification). [Online] 01/2010.
<http://www.rfc-editor.org/rfc/rfc5751.txt>
- [6] RSA Laboratories - Standards Initiatives - Public-Key Cryptography Standards (PKCS). [Online] 15. február 2011.
<http://www.rsa.com/rsalabs/node.asp?id=2124>
- [7] RSA Laboratories: PKCS #11: Base Functionality v2.30: Cryptoki – Draft 4. [Online] 10. júl. 2009.
<ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-30/pkcs-11v2-30b-d6.pdf>
- [8] RSA. *Wikipedia, the free encyclopedia*. [Online] 18. február 2011.
<http://en.wikipedia.org/wiki/Rsa>
- [9] PALÚCH, S.: *Jednocestné hashovacie funkcie* (prednáška z predmetu Kryptografia a bezpečnosť). Žilina, 2010.
- [10] Base64. *Wikipedia, the free encyclopedia*. [Online] 23. február 2011.
<http://en.wikipedia.org/wiki/Base64>
- [11] Abstract Syntax Notation One. *Wikipedia, the free encyclopedia*. [Online] 23. február 2011.
http://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One
- [12] Burton S. Kaliski Jr.: A Layman's Guide to a Subset of ASN.1, BER and DER, RSA Laboratories. Redwood City, CA, USA. 1. november 1993.
- [13] RSA Laboratories: PKCS#6: Extended-Certificate Syntax Standard. [Online] 1. november 1993.
<ftp://ftp.rsasecurity.com/pub/pkcs/ascii/pkcs-6.asc>

- [14] RSA Laboratories: PKCS#8: Private-Key Information Syntax Standard. [Online] 1. november 1993.
<ftp://ftp.rsa.com/pub/pkcs/ascii/pkcs-8.asc>
- [15] D. Balfanz, E. W. Felten: Hand-Held Computers Can Be Better Smart Cards, Proceeding, SSYM'99 Proceedings of the 8th conference on USENIX Security Symposium - Volume 8, USENIX Association Berkeley, CA, USA, 1999.
- [16] Metodické usmernenie 14/2009-R Ministerstva školstva SR o náležitostiach záverečných prác, ich bibliografickej registrácii, kontrole originality, uchovávaní a sprístupňovaní, č. CD-2009–31655/30400-1:02, 27. august 2009.
- [17] HITTMÁR, Š.: *Metodická pomôcka pre spracovateľov záverečných prác bakalárskeho štúdia - bakalárskych prác*. Žilina, 2007.

10.3 ZOZNAM PRÍLOH

- Príloha č. 1: Zoznam PKCS#11 funkcií
- Príloha č. 2: Diagram tried projektu CrySign Mobile
- Príloha č. 3: Diagram tried projektu PKCS11Lib
- Príloha č. 4: Diagram tried projektu PInvokeLib
- Príloha č. 5: Volania PKCS#11 funkcií pri spustení PKCS#11 programu
- Príloha č. 6: Volania PKCS#11 funkcií pri zobrazení zoznamu certifikátov
- Príloha č. 7: Volania PKCS#11 funkcií pri odoslaní digitálne podpísaného e-mailu
- Príloha č. 8: Volania PKCS#11 funkcií pri čítaní zašifrovaného e-mailu
- Príloha č. 9: Volania PKCS#11 funkcií pri prihlásení pomocou certifikátu na webe
- Príloha č. 10: Volania PKCS#11 funkcií pri generovaní RSA kľúčov
- Príloha č. 11: DVD so zdrojovými kódmi i dokumentáciou

11 PRÍLOHY

Príloha č. 1

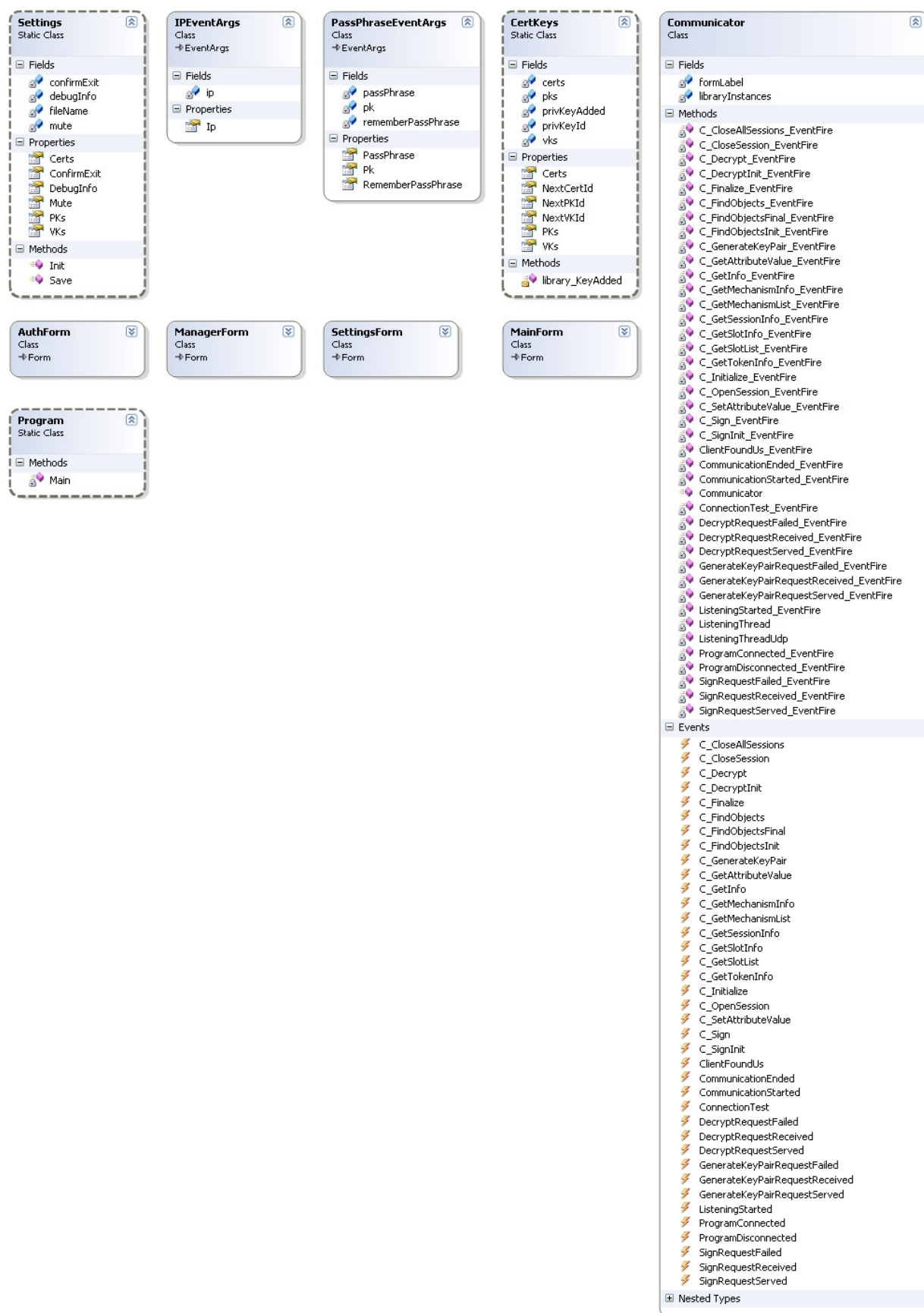
Zaradenie	Funkcia	Popis
Všeobecné funkcie	C_Initialize	Inicializuje modul
	C_Finalize	Dealokuje prostriedky
	C_GetInfo	Poskytne všeobecné informácie o module
	C_GetFunctionList	Poskytne vstupné body všetkých funkcií v module
Funkcie pre správu slotov a tokenov	C_GetSlotList	Poskytne zoznam všetkých slotov
	C_GetSlotInfo	Poskytne informácie o danom slotе
	C_GetTokenInfo	Poskytne informácie o danom tokene
	C_WaitForSlotEvent	Počká na určitú udalosť, kým nastane na danom slotе
	C_GetMechanismList	Vráti zoznam mechanizmov, ktoré podporuje daný slot
	C_GetMechanismInfo	Vráti informácie o danom mechanizme
	C_InitToken	Inicializuje token
	C_InitPIN	Inicializuje PIN bežného užívateľa
	C_SetPIN	Zmení PIN terajšieho užívateľa
Funkcie pre manipuláciu s reláciami	C_OpenSession	Otvorí reláciu
	C_CloseSession	Ukončí reláciu
	C_CloseAllSessions	Ukončí všetky relácie otvorené s určitým tokenom
	C_GetSessionInfo	Poskytne informácie o danej relácii
	C_GetOperationState	Poskytne stav kryptografickej operácie

	C_SetOperationState	Nastaví stav kryptografickej operácie
	C_Login	Prihlási sa do tokenu
	C_Logout	Odhlási sa z tokenu
Funkcie pre manipuláciu s objektmi	C_CreateObject	Vytvorí objekt
	C_CopyObject	Vytvorí kópiu objektu
	C_DestroyObject	Odstráni objekt
	C_GetObjectSize	Vráti veľkosť objektu v bajtoch
	C_GetAttributeValue	Vráti hodnotu atribútu objektu
	C_SetAttributeValue	Nastaví hodnotu atribútu objektu
	C_FindObjectsInit	Inicializuje operáciu vyhľadávania objektov
	C_FindObjects	Uskutoční či pokračuje vo vyhľadávaní objektov
	C_FindObjectsFinal	Ukončí vyhľadávanie objektov
Šifrovacie funkcie	C_EncryptInit	Inicializuje operáciu šifrovania
	C_Encrypt	Zašifruje dáta na jeden raz
	C_EncryptUpdate	Pokračuje vo viacnásobnom šifrovaní
	C_EncryptFinal	Ukončí viacnásobné šifrovanie
Dešifrovacie funkcie	C_DecryptInit	Inicializuje operáciu dešifrovania
	C_Decrypt	Dešifruje dáta na jeden raz
	C_DecryptUpdate	Pokračuje vo viacnásobnom dešifrovaní
	C_DecryptFinal	Ukončí viacnásobné dešifrovanie
MD funkcie	C_DigestInit	Inicializuje MD operáciu
	C_Digest	Vykoná MD operáciu na jeden raz
	C_DigestUpdate	Pokračuje vo viacnásobnej MD operácii

	C_DigestKey	Vykoná MD operáciu nad kľúčom
	C_DigestFinal	Ukončí viacnásobnú MD operáciu
Funkcie na digitálne podpisovanie	C_SignInit	Inicializuje podpisovanie
	C_Sign	Podpíše dáta na jeden raz
	C_SignUpdate	Pokračuje vo viacnásobnom podpisovaní
	C_SignFinal	Ukončí viacnásobné podpisovanie
	C_SignRecoverInit	Inicializuje podpisovanie, pri ktorom sa dáta dajú obnoviť z podpisu
	C_SignRecover	Podpíše dáta tak, aby sa dali obnoviť z podpisu
Funkcie na overovanie digitálneho podpisu	C_VerifyInit	Inicializuje overenie podpisu
	C_Verify	Overí podpis na jeden raz
	C_VerifyUpdate	Pokračuje vo viacnásobnom overovaní podpisu
	C_VerifyFinal	Ukončí viacnásobné overovanie podpisu
	C_VerifyRecoverInit	Inicializuje overovanie podpisu, kde dáta sú získané z podpisu
	C_VerifyRecover	Overí podpis, pričom dáta sú získané z podpisu
Dvojúčelové funkcie	C_DigestEncryptUpdate	Pokračuje v súčasnom vykonávaní MD operácie a šifrovania
	C_DecryptDigestUpdate	Pokračuje v súčasnom vykonávaní MD operácie a dešifrovania
	C_SignEncryptUpdate	Pokračuje vo súčasnom podpisovaní a šifrovaní
	C_DecryptVerifyUpdate	Pokračuje v súčasnom dešifrovaní a overení podpisu

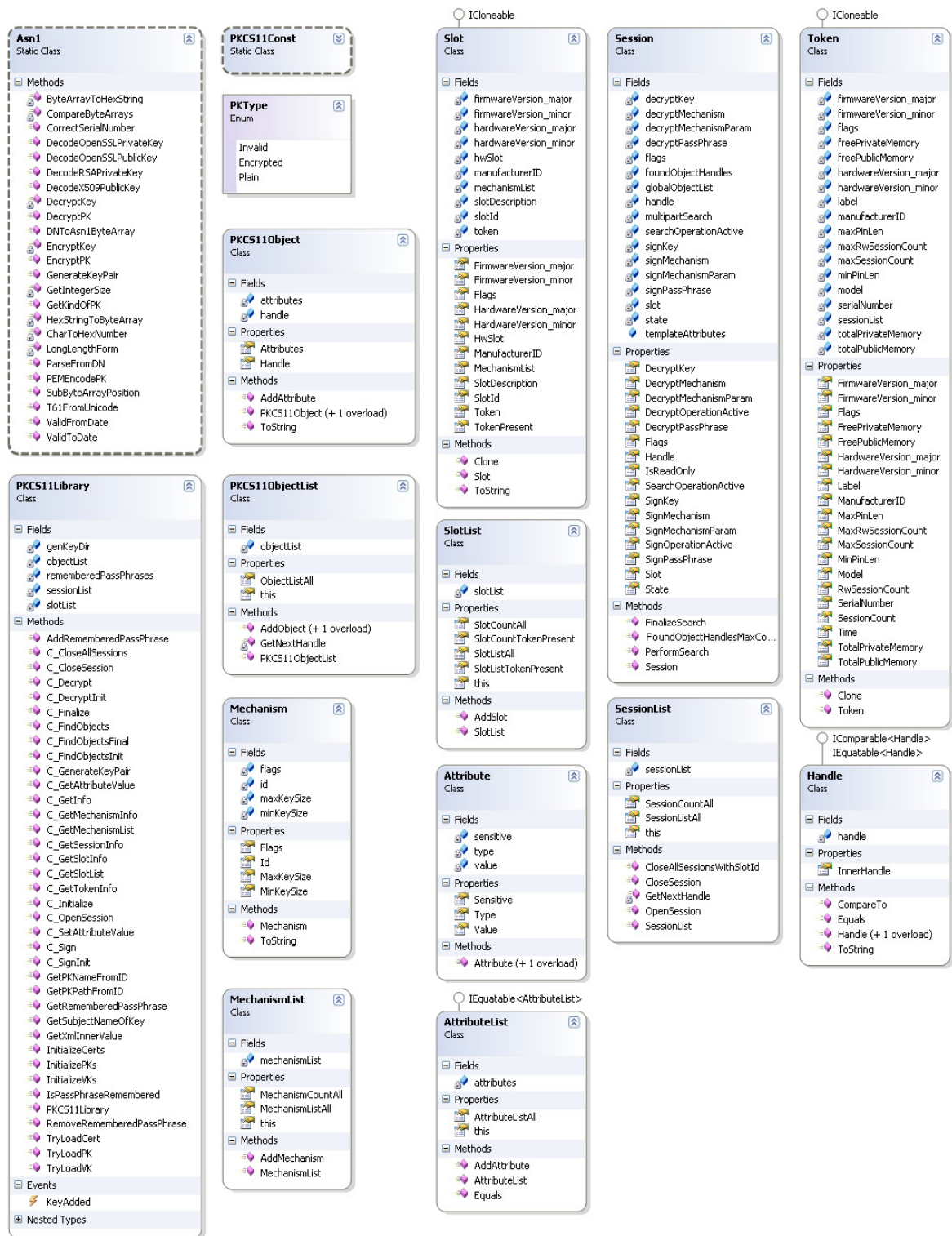
Funkcie správy kľúčov	C_GenerateKey	Vygeneruje symetrický kľúč
	C_GenerateKeyPair	Vygeneruje pár kľúčov (súkromný + verejný kľúč)
	C_WrapKey	Zašifruje kľúč
	C_UnwrapKey	Dešifruje kľúč
	C_DeriveKey	Odvodí kľúč z iného kľúča
Generovanie náhodných čísel	C_SeedRandom	Poskytne nový materiál pre násadu generátora náh. čísel
	C_GenerateRandom	Vygeneruje náhodné dáta
Podpora paralelizmu	C_GetFunctionStatus	Zastaraná funkcia vracajúca CKR_FUNCTION_NOT_PARALLEL
	C_CancelFunction	Zastaraná funkcia vracajúca CKR_FUNCTION_NOT_PARALLEL

Príloha č. 2



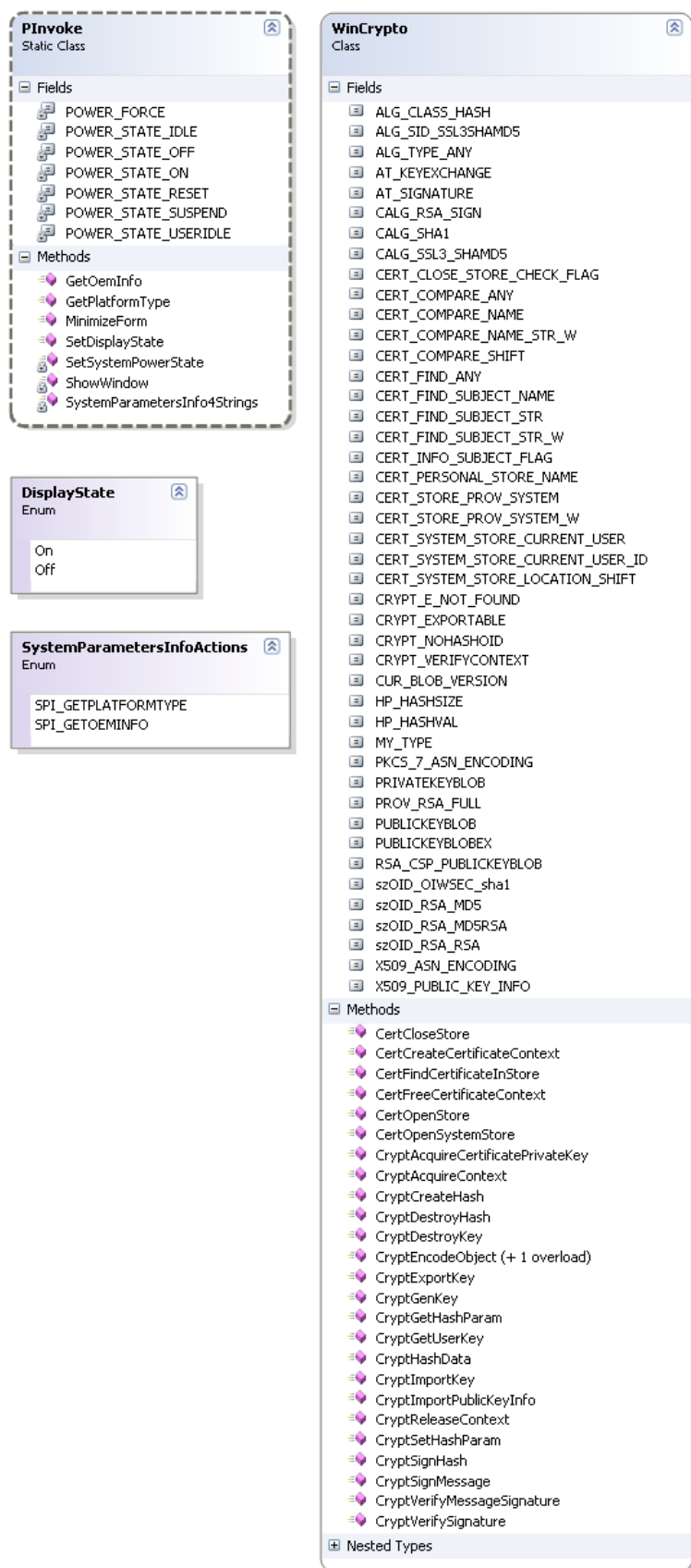
Obr. 11-1: Diagram tried projektu *CrySign Mobile*

Priloha č. 3



Obr. 11-2: Diagram tried projektu *PKCS11Lib*

Príloha č. 4



Obr. 11-3: Diagram tried projektu *PInvokeLib*

Príloha č. 5

```
0: C_GetFunctionList
Returned: 0 CKR_OK

1: C_Initialize
Returned: 10 CKR_CANT_LOCK

2: C_Initialize
Returned: 0 CKR_OK

3: C_GetInfo
    cryptokiVersion:      2.30
    manufacturerID:       'Matej Kurpel'
    flags:                 0
    libraryDescription:    'WMPKCS#11 - WinMobile module'
    libraryVersion:        1.0
Returned: 0 CKR_OK

4: C_GetSlotList
[in] tokenPresent = 0x0
[out] pSlotList:
Count is 1
[out] *pulCount = 0x1
Returned: 0 CKR_OK

5: C_GetSlotList
[in] tokenPresent = 0x0
[out] pSlotList:
Slot 0
[out] *pulCount = 0x1
Returned: 0 CKR_OK

6: C_GetSlotInfo
[in] slotID = 0x0
[out] pInfo:
    slotDescription:       'Mobile Device Slot'
    manufacturerID:        'Unknown Manufacturer'
    hardwareVersion:        1.0
    firmwareVersion:        1.0
    flags:                  5
    CKF_TOKEN_PRESENT
    CKF_HW_SLOT
Returned: 0 CKR_OK

7: C_GetTokenInfo
[in] slotID = 0x0
[out] pInfo:
    label:                 'HTC Touch HD T8282'
    manufacturerID:         'Unknown Manufacturer'
    model:                  'Unknown Model'
    serialNumber:           'Unknown S/N'
    ulMaxSessionCount:      0
    ulSessionCount:         0
    ulMaxRwSessionCount:    1
    ulRwSessionCount:       0
    ulMaxPinLen:            32
    ulMinPinLen:            1
    ulTotalPublicMemory:    -1
    ulFreePublicMemory:     -1
    ulTotalPrivateMemory:   -1
    ulFreePrivateMemory:    -1
    hardwareVersion:        1.0
    firmwareVersion:        1.0
    time:                   '2011030112172600'
```

```

        flags:                                548
        CKF_USER_PIN_INITIALIZED
        CKF_CLOCK_ON_TOKEN
        CKF_PROTECTED_AUTHENTICATION_PATH
        CKF_TOKEN_INITIALIZED
Returned:  0 CKR_OK

8: C_GetMechanismList
[in] slotID = 0x0
[out] pMechanismList[2]:
Count is 2
Returned:  0 CKR_OK

9: C_GetMechanismList
[in] slotID = 0x0
[out] pMechanismList[2]:
    CKM_RSA_PKCS
    CKM_RSA_PKCS_KEY_PAIR_GEN
Returned:  0 CKR_OK

10: C_OpenSession
[in] slotID = 0x0
[in] flags = 0x4
pApplication=066F8C00
Notify=6724A378
[out] *phSession = 0x1
Returned:  0 CKR_OK

11: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[1]:
    CKA_CLASS                                CKO_NETSCAPE_BUILTIN_ROOT_LIST
Returned:  19 CKR_ATTRIBUTE_VALUE_INVALID

12: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_TOKEN                                True
    CKA_CLASS                                CKO_CERTIFICATE
Returned:  0 CKR_OK

13: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0xa
[out] ulObjectCount = 0x1
Object 1 Matches
Returned:  0 CKR_OK

14: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

15: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_TOKEN                                requested with 0 buffer
    CKA_LABEL                                requested with 0 buffer
[out] pTemplate[2]:
    CKA_TOKEN                                has size 1
    CKA_LABEL                                has size 12
Returned:  0 CKR_OK

16: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_TOKEN                                requested with 1 buffer

```

```

    CKA_LABEL                requested with 12 buffer
[out] pTemplate[2]:
    CKA_TOKEN                True
    CKA_LABEL                [size : 0xC (12)]
    4D617465 6A204B75 7270656C
    M a t e j . K u r p e l
Returned: 0 CKR_OK
17: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[10]:
    CKA_CLASS                requested with 0 buffer
    CKA_TOKEN                requested with 0 buffer
    CKA_LABEL                requested with 0 buffer
    CKA_CERTIFICATE_TYPE    requested with 0 buffer
    CKA_ID                   requested with 0 buffer
    CKA_VALUE                requested with 0 buffer
    CKA_ISSUER               requested with 0 buffer
    CKA_SERIAL_NUMBER        requested with 0 buffer
    CKA_SUBJECT              requested with 0 buffer
    CKA_NETSCAPE_EMAIL(Netsc) requested with 0 buffer
[out] pTemplate[10]:
    CKA_CLASS                has size 4
    CKA_TOKEN                has size 1
    CKA_LABEL                has size 12
    CKA_CERTIFICATE_TYPE    has size 4
    CKA_ID                   has size 4
    CKA_VALUE                has size 1311
    CKA_ISSUER               has size 19
    CKA_SERIAL_NUMBER        has size 12
    CKA_SUBJECT              has size 133
    CKA_NETSCAPE_EMAIL(Netsc) has size -1
Returned: 18 CKR_ATTRIBUTE_TYPE_INVALID

18: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[10]:
    CKA_CLASS                requested with 4 buffer
    CKA_TOKEN                requested with 1 buffer
    CKA_LABEL                requested with 12 buffer
    CKA_CERTIFICATE_TYPE    requested with 4 buffer
    CKA_ID                   requested with 4 buffer
    CKA_VALUE                requested with 1311 buffer
    CKA_ISSUER               requested with 19 buffer
    CKA_SERIAL_NUMBER        requested with 12 buffer
    CKA_SUBJECT              requested with 133 buffer
    CKA_NETSCAPE_EMAIL(Netsc) requested with 0 buffer
[out] pTemplate[10]:
    CKA_CLASS                CKO_CERTIFICATE
    CKA_TOKEN                True
    CKA_LABEL                [size : 0xC (12)]
    4D617465 6A204B75 7270656C
    M a t e j . K u r p e l
    CKA_CERTIFICATE_TYPE    CKC_X_509
    CKA_ID                   [size : 0x4 (4)]
    01000000
    CKA_VALUE                [size : 0x51F (1311)]
    3082051B 30820403 A0030201 ..... (skrátene)
    CKA_ISSUER               [size : 0x13 (19)]
    3011310F 300D0603 55040313 06452D43 657274
    DN: CN=E-Cert
    CKA_SERIAL_NUMBER        [size : 0xC (12)]
    020A7F02 EF4B0000 00000E87
    CKA_SUBJECT              [size : 0x85 (133)]
    30818231 1E301C06 0355040A ..... (skrátene)
    DN: O=\xC5\xBDilinsk\xC3\xA1 univerzita, OU=Fakulta riadenia a informatiky, CN=Matej
Kurpel/emailAddress=mkurpel@gmail.com

```



```

    CKA_NETSCAPE_EMAIL(Netsc)                has size -1
Returned:  18 CKR_ATTRIBUTE_TYPE_INVALID

19: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[1]:
    CKA_NETSCAPE_EMAIL(Netsc)                requested with 0 buffer
[out] pTemplate[1]:
    CKA_NETSCAPE_EMAIL(Netsc)                has size -1
Returned:  18 CKR_ATTRIBUTE_TYPE_INVALID

20: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[1]:
    CKA_NETSCAPE_EMAIL(Netsc)                requested with 0 buffer
[out] pTemplate[1]:
    CKA_NETSCAPE_EMAIL(Netsc)                has size -1
Returned:  18 CKR_ATTRIBUTE_TYPE_INVALID

21: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_TOKEN                                True
    CKA_CLASS                                CKO_NETSCAPE_TRUST
Returned:  19 CKR_ATTRIBUTE_VALUE_INVALID

22: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_CLASS                                CKO_NETSCAPE_CRL
    CKA_SUBJECT                                [size : 0x48 (72)]
    3046310B 30090603 55040613 02555331 13301106 0355040A 130A476F 6F676C65
    20496E63 31223020 06035504 03131947 6F6F676C 6520496E 7465726E 65742041
    7574686F 72697479
    DN: C=US, O=Google Inc, CN=Google Internet Authority
Returned:  19 CKR_ATTRIBUTE_VALUE_INVALID

23: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_CLASS                                CKO_NETSCAPE_CRL
    CKA_SUBJECT                                [size : 0x50 (80)]
    304E310B 30090603 55040613 02555331 10300E06 0355040A 13074571 75696661
    78312D30 2B060355 040B1324 45717569 66617820 53656375 72652043 65727469
    66696361 74652041 7574686F 72697479
    DN: C=US, O=Equifax, OU=Equifax Secure Certificate Authority
Returned:  19 CKR_ATTRIBUTE_VALUE_INVALID

```

Príloha č. 6

```
31: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_TOKEN          True
    CKA_CLASS          CKO_CERTIFICATE
Returned: 0 CKR_OK

32: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x10
[out] ulObjectCount = 0x1
Object 1 Matches
Returned: 0 CKR_OK

33: C_FindObjectsFinal
[in] hSession = 0x1
Returned: 0 CKR_OK

34: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_TOKEN          requested with 0 buffer
    CKA_LABEL          requested with 0 buffer
[out] pTemplate[2]:
    CKA_TOKEN          has size 1
    CKA_LABEL          has size 12
Returned: 0 CKR_OK

35: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_TOKEN          requested with 1 buffer
    CKA_LABEL          requested with 12 buffer
[out] pTemplate[2]:
    CKA_TOKEN          True
    CKA_LABEL          [size : 0xC (12)]
    4D617465 6A204B75 7270656C
    M a t e j . K u r p e l
Returned: 0 CKR_OK

36: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID             requested with 0 buffer
    CKA_CLASS          requested with 0 buffer
[out] pTemplate[2]:
    CKA_ID             has size 4
    CKA_CLASS          has size 4
Returned: 0 CKR_OK

37: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID             requested with 4 buffer
    CKA_CLASS          requested with 4 buffer
[out] pTemplate[2]:
    CKA_ID             [size : 0x4 (4)]
    01000000
    CKA_CLASS          CKO_CERTIFICATE
Returned: 0 CKR_OK
```

```

38: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_ID                [size : 0x4 (4)]
    01000000
    CKA_CLASS              CKO_PRIVATE_KEY
Returned:  0 CKR_OK

39: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 2 Matches
Returned:  0 CKR_OK

40: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

```

Príloha č. 7

```
28: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID            requested with 0 buffer
    CKA_CLASS          requested with 0 buffer
[out] pTemplate[2]:
    CKA_ID            has size 4
    CKA_CLASS          has size 4
Returned:  0 CKR_OK

29: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID            requested with 4 buffer
    CKA_CLASS          requested with 4 buffer
[out] pTemplate[2]:
    CKA_ID            [size : 0x4 (4)]
    01000000
    CKA_CLASS          CKO_CERTIFICATE
Returned:  0 CKR_OK

30: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_ID            [size : 0x4 (4)]
    01000000
    CKA_CLASS          CKO_PRIVATE_KEY
Returned:  0 CKR_OK

31: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 2 Matches
Returned:  0 CKR_OK

32: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

33: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_CLASS          CKO_NETSCAPE_CRL
    CKA_SUBJECT          [size : 0x13 (19)]
    3011310F 300D0603 55040313 06452D43 657274
    DN: CN=E-Cert
Returned:  19 CKR_ATTRIBUTE_VALUE_INVALID

34: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[1]:
    CKA_VALUE          [size : 0x51F (1311)]
    3082051B 30820403 A0030201 ..... (skrátené)
Returned:  0 CKR_OK

35: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 1 Matches
Returned:  0 CKR_OK
```

```

36: C_FindObjectsFinal
[in] hSession = 0x1
Returned: 0 CKR_OK

37: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID          requested with 0 buffer
    CKA_CLASS       requested with 0 buffer
[out] pTemplate[2]:
    CKA_ID          has size 4
    CKA_CLASS       has size 4
Returned: 0 CKR_OK

38: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID          requested with 4 buffer
    CKA_CLASS       requested with 4 buffer
[out] pTemplate[2]:
    CKA_ID          [size : 0x4 (4)]
    01000000
    CKA_CLASS       CKO_CERTIFICATE
Returned: 0 CKR_OK

39: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_ID          [size : 0x4 (4)]
    01000000
    CKA_CLASS       CKO_PRIVATE_KEY
Returned: 0 CKR_OK

40: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 2 Matches
Returned: 0 CKR_OK

41: C_FindObjectsFinal
[in] hSession = 0x1
Returned: 0 CKR_OK

42: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_KEY_TYPE    requested with 4 buffer
[out] pTemplate[1]:
    CKA_KEY_TYPE    CKK_RSA
Returned: 0 CKR_OK

43: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_TOKEN       requested with 1 buffer
[out] pTemplate[1]:
    CKA_TOKEN       True
Returned: 0 CKR_OK

44: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:

```

```

        CKA_PRIVATE                requested with 1 buffer
[out] pTemplate[1]:
        CKA_PRIVATE                True
Returned:  0 CKR_OK

45: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
        CKA_MODULUS                requested with 0 buffer
[out] pTemplate[1]:
        CKA_MODULUS                has size 256
Returned:  0 CKR_OK

46: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
        CKA_MODULUS                requested with 256 buffer
[out] pTemplate[1]:
        CKA_MODULUS                [size : 0x100 (256)]
        A6CC2DB3 B1333AEA C99419FD ..... (skrátené)
Returned:  0 CKR_OK

47: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
        CKA_PRIVATE                requested with 1 buffer
[out] pTemplate[1]:
        CKA_PRIVATE                True
Returned:  0 CKR_OK

48: C_OpenSession
[in] slotID = 0x0
[in] flags = 0x4
pApplication=066F0000
Notify=5B4AA378
[out] *phSession = 0x2
Returned:  0 CKR_OK

49: C_SignInit
[in] hSession = 0x2
pMechanism->type=CKM_RSA_PKCS
[in] hKey = 0x2
Returned:  0 CKR_OK

50: C_Sign
[in] hSession = 0x2
[in] pData[ulDataLen] [size : 0x23 (35)]
        30213009 06052B0E 03021A05 00041452 0DA8E820 BB7BC6F7 04DFA9E7 83801D8B
        F6961E
[out] pSignature[*pulSignatureLen] [size : 0x100 (256)]
        7F8677F8 35CAD945 E872A1E4 ..... (skrátené)
Returned:  0 CKR_OK

51: C_CloseSession
[in] hSession = 0x2
Returned:  0 CKR_OK

52: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
        CKA_CLASS                  CKO_NETSCAPE_CRL
        CKA_SUBJECT                  [size : 0x48 (72)]
        3046310B 30090603 55040613 02555331 13301106 0355040A 130A476F 6F676C65
        20496E63 31223020 06035504 03131947 6F6F676C 6520496E 7465726E 65742041
        7574686F 72697479

```

DN: C=US, O=Google Inc, CN=Google Internet Authority
Returned: 19 CKR_ATTRIBUTE_VALUE_INVALID

53: C_FindObjectsInit

[in] hSession = 0x1

[in] pTemplate[2]:

CKA_CLASS CKO_NETSCAPE_CRL

CKA_SUBJECT [size : 0x50 (80)]

304E310B 30090603 55040613 02555331 10300E06 0355040A 13074571 75696661

78312D30 2B060355 040B1324 45717569 66617820 53656375 72652043 65727469

66696361 74652041 7574686F 72697479

DN: C=US, O=Equifax, OU=Equifax Secure Certificate Authority

Returned: 19 CKR_ATTRIBUTE_VALUE_INVALID

54: C_FindObjectsInit

[in] hSession = 0x1

[in] pTemplate[3]:

CKA_SUBJECT [size : 0x85 (133)]

30818231 1E301C06 0355040A 0C15C5BD 696C696E 736BC3A1 20756E69 7665727A

69746131 27302506 0355040B 131E4661 6B756C74 61207269 6164656E 69612061

20696E66 6F726D61 74696B79 31153013 06035504 03130C4D 6174656A 204B7572

70656C31 20301E06 092A8648 86F70D01 09011611 6D6B7572 70656C40 676D6169

6C2E636F 6D

DN: O=\xC5\xBDilinsk\xC3\xA1 univerzita, OU=Fakulta riadenia a informatiky, CN=Matej

Kurpel/emailAddress=mkurpel@gmail.com

CKA_CLASS CKO_NETSCAPE_SMIME

CKA_NETSCAPE_EMAIL(Netsc) [size : 0x11 (17)]

6D6B7572 70656C40 676D6169 6C2E636F 6D

Returned: 19 CKR_ATTRIBUTE_VALUE_INVALID

Príloha 8

```
30: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[4]:
    CKA_TOKEN          True
    CKA_CLASS           CKO_CERTIFICATE
    CKA_ISSUER          [size : 0x13 (19)]
    3011310F 300D0603 55040313 06452D43 657274
    DN: CN=E-Cert
    CKA_SERIAL_NUMBER   [size : 0xC (12)]
    020A7F02 EF4B0000 00000E87
Returned:  0 CKR_OK

31: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 1 Matches
Returned:  0 CKR_OK

32: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

33: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_TOKEN          requested with 0 buffer
    CKA_LABEL           requested with 0 buffer
[out] pTemplate[2]:
    CKA_TOKEN          has size 1
    CKA_LABEL          has size 12
Returned:  0 CKR_OK

34: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_TOKEN          requested with 1 buffer
    CKA_LABEL           requested with 12 buffer
[out] pTemplate[2]:
    CKA_TOKEN          True
    CKA_LABEL           [size : 0xC (12)]
    4D617465 6A204B75 7270656C
    M a t e j . K u r p e l
Returned:  0 CKR_OK

35: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID              requested with 0 buffer
    CKA_CLASS           requested with 0 buffer
[out] pTemplate[2]:
    CKA_ID              has size 4
    CKA_CLASS           has size 4
Returned:  0 CKR_OK

36: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID              requested with 4 buffer
    CKA_CLASS           requested with 4 buffer
[out] pTemplate[2]:
```



```

        CKA_ID                [size : 0x4 (4)]
        01000000
        CKA_CLASS              CKO_CERTIFICATE
Returned:  0 CKR_OK

37: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
        CKA_ID                [size : 0x4 (4)]
        01000000
        CKA_CLASS              CKO_PRIVATE_KEY
Returned:  0 CKR_OK

38: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 2 Matches
Returned:  0 CKR_OK

39: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

40: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[1]:
        CKA_VALUE              [size : 0x51F (1311)]
        3082051B 30820403 A0030201 ..... (skrátené)
Returned:  0 CKR_OK

41: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 1 Matches
Returned:  0 CKR_OK

42: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

43: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
        CKA_ID                requested with 0 buffer
        CKA_CLASS              requested with 0 buffer
[out] pTemplate[2]:
        CKA_ID                has size 4
        CKA_CLASS              has size 4
Returned:  0 CKR_OK

44: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
        CKA_ID                requested with 4 buffer
        CKA_CLASS              requested with 4 buffer
[out] pTemplate[2]:
        CKA_ID                [size : 0x4 (4)]
        01000000
        CKA_CLASS              CKO_CERTIFICATE
Returned:  0 CKR_OK

```

```

45: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_ID                [size : 0x4 (4)]
    01000000
    CKA_CLASS              CKO_PRIVATE_KEY
Returned:  0 CKR_OK

46: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 2 Matches
Returned:  0 CKR_OK

47: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

48: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_KEY_TYPE           requested with 4 buffer
[out] pTemplate[1]:
    CKA_KEY_TYPE           CKK_RSA
Returned:  0 CKR_OK

49: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_TOKEN              requested with 1 buffer
[out] pTemplate[1]:
    CKA_TOKEN              True
Returned:  0 CKR_OK

50: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_PRIVATE            requested with 1 buffer
[out] pTemplate[1]:
    CKA_PRIVATE            True
Returned:  0 CKR_OK

51: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_PRIVATE            requested with 1 buffer
[out] pTemplate[1]:
    CKA_PRIVATE            True
Returned:  0 CKR_OK

52: C_OpenSession
[in] slotID = 0x0
[in] flags = 0x4
pApplication=06BEC400
Notify=5940A378
[out] *phSession = 0x2
Returned:  0 CKR_OK

53: C_DecryptInit
[in] hSession = 0x2
pMechanism->type=CKM_RSA_PKCS
[in] hKey = 0x2
Returned:  0 CKR_OK

```

```

54: C_Decrypt
[in] hSession = 0x2
[in] pEncryptedData[ulEncryptedDataLen] [size : 0x100 (256)]
    A0820771 021EA0A8 BC2203CB 7C0F8289 5942CC92 09404F18 95F32D44 710CA1A1
    1DDAA5F2 0DB6586D 72ADCE27 3FEAEE7B BA6E8FC9 75331D09 AB80BFF4 6BC75C21
    3AC57944 2DF3CE2D 769F40C0 073D0CC5 880E6780 46C3BBBF 7764E5E5 D349358B
    B1858CEB 0CCFD536 4AB0F9A8 8D75F9FB 14FFE21E 038D4D70 D9B7170A E7CC6BE1
    CDC48B15 9A0E10DF 9B22139C 0A1C013E 8F66A8BB 002133C4 83F400E1 6D1BB8F2
    ED50D715 E86EFB49 BD34695A C2DE6006 D306D863 9FD6C23F E428BC3D 26093C7E
    F54D3FE5 BFA093FE 1F878780 4A3CBAB6 ADC7B204 D0C3E8C2 5702771F 2927333B
    591F3C61 3B57A8F6 E51738E0 1D921334 C17A66BA 369AB08E 76F5A06E BF2CAD4E
[out] pData[*pulDataLen] [size : 0x18 (24)]
    E59BA731 E65D3804 3E5D1029 40680DB5 5BE9E3DC 5ED3CDCB
Returned:  0 CKR_OK

55: C_CloseSession
[in] hSession = 0x2
Returned:  0 CKR_OK

```

Príloha 9

```
23: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_TOKEN          True
    CKA_CLASS          CKO_CERTIFICATE
Returned: 0 CKR_OK

24: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x10
[out] ulObjectCount = 0x1
Object 1 Matches
Returned: 0 CKR_OK

25: C_FindObjectsFinal
[in] hSession = 0x1
Returned: 0 CKR_OK

26: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_TOKEN          requested with 0 buffer
    CKA_LABEL          requested with 0 buffer
[out] pTemplate[2]:
    CKA_TOKEN          has size 1
    CKA_LABEL          has size 12
Returned: 0 CKR_OK

27: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_TOKEN          requested with 1 buffer
    CKA_LABEL          requested with 12 buffer
[out] pTemplate[2]:
    CKA_TOKEN          True
    CKA_LABEL          [size : 0xC (12)]
    4D617465 6A204B75 7270656C
    M a t e j . K u r p e l
Returned: 0 CKR_OK

28: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID             requested with 0 buffer
    CKA_CLASS          requested with 0 buffer
[out] pTemplate[2]:
    CKA_ID             has size 4
    CKA_CLASS          has size 4
Returned: 0 CKR_OK

29: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID             requested with 4 buffer
    CKA_CLASS          requested with 4 buffer
[out] pTemplate[2]:
    CKA_ID             [size : 0x4 (4)]
    01000000
    CKA_CLASS          CKO_CERTIFICATE
Returned: 0 CKR_OK
```

```

30: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_ID                [size : 0x4 (4)]
    01000000
    CKA_CLASS              CKO_PRIVATE_KEY
Returned:  0 CKR_OK

31: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 2 Matches
Returned:  0 CKR_OK

32: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

33: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID                requested with 0 buffer
    CKA_CLASS              requested with 0 buffer
[out] pTemplate[2]:
    CKA_ID                has size 4
    CKA_CLASS              has size 4
Returned:  0 CKR_OK

34: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID                requested with 4 buffer
    CKA_CLASS              requested with 4 buffer
[out] pTemplate[2]:
    CKA_ID                [size : 0x4 (4)]
    01000000
    CKA_CLASS              CKO_CERTIFICATE
Returned:  0 CKR_OK

35: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_ID                [size : 0x4 (4)]
    01000000
    CKA_CLASS              CKO_PRIVATE_KEY
Returned:  0 CKR_OK

36: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 2 Matches
Returned:  0 CKR_OK

37: C_FindObjectsFinal
[in] hSession = 0x1
Returned:  0 CKR_OK

38: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[1]:
    CKA_VALUE              [size : 0x51F (1311)]
    3082051B 30820403 A0030201 ..... (skrátene)
Returned:  0 CKR_OK

```

```

39: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 1 Matches
Returned: 0 CKR_OK

40: C_FindObjectsFinal
[in] hSession = 0x1
Returned: 0 CKR_OK

41: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID          requested with 0 buffer
    CKA_CLASS        requested with 0 buffer
[out] pTemplate[2]:
    CKA_ID          has size 4
    CKA_CLASS        has size 4
Returned: 0 CKR_OK

42: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x1
[in] pTemplate[2]:
    CKA_ID          requested with 4 buffer
    CKA_CLASS        requested with 4 buffer
[out] pTemplate[2]:
    CKA_ID          [size : 0x4 (4)]
    01000000
    CKA_CLASS        CKO_CERTIFICATE
Returned: 0 CKR_OK

43: C_FindObjectsInit
[in] hSession = 0x1
[in] pTemplate[2]:
    CKA_ID          [size : 0x4 (4)]
    01000000
    CKA_CLASS        CKO_PRIVATE_KEY
Returned: 0 CKR_OK

44: C_FindObjects
[in] hSession = 0x1
[in] ulMaxObjectCount = 0x1
[out] ulObjectCount = 0x1
Object 2 Matches
Returned: 0 CKR_OK

45: C_FindObjectsFinal
[in] hSession = 0x1
Returned: 0 CKR_OK

46: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_KEY_TYPE     requested with 4 buffer
[out] pTemplate[1]:
    CKA_KEY_TYPE     CKK_RSA
Returned: 0 CKR_OK

47: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_TOKEN        requested with 1 buffer
[out] pTemplate[1]:

```

```

        CKA_TOKEN                True
Returned:  0 CKR_OK

48: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_PRIVATE                requested with 1 buffer
[out] pTemplate[1]:
    CKA_PRIVATE                True
Returned:  0 CKR_OK

49: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_MODULUS                requested with 0 buffer
[out] pTemplate[1]:
    CKA_MODULUS                has size 256
Returned:  0 CKR_OK

50: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_MODULUS                requested with 256 buffer
[out] pTemplate[1]:
    CKA_MODULUS                [size : 0x100 (256)]
    A6CC2DB3 B1333AEA C99419FD ..... (skrátané)
Returned:  0 CKR_OK

51: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x2
[in] pTemplate[1]:
    CKA_PRIVATE                requested with 1 buffer
[out] pTemplate[1]:
    CKA_PRIVATE                True
Returned:  0 CKR_OK

52: C_OpenSession
[in] slotID = 0x0
[in] flags = 0x4
pApplication=0759EC00
Notify=6C5DA378
[out] *phSession = 0x2
Returned:  0 CKR_OK

53: C_SignInit
[in] hSession = 0x2
pMechanism->type=CKM_RSA_PKCS
[in] hKey = 0x2
Returned:  0 CKR_OK

54: C_Sign
[in] hSession = 0x2
[in] pData[ulDataLen] [size : 0x24 (36)]
    4011FD81 5D93A85E 7FF26E14 6575274D D3BED65B 70833926 B451B842 1B57AC78
    172D0D2D
[out] pSignature[*pSignatureLen] [size : 0x100 (256)]
    899AA3D6 F558BBE6 6E06DDF3 ..... (skrátané)
Returned:  0 CKR_OK

55: C_CloseSession
[in] hSession = 0x2
Returned:  0 CKR_OK

```

Príloha 10

```
12: C_GetMechanismInfo
[in] slotID = 0x0
CKM_RSA_PKCS
[out] pInfo:
CKM_RSA_PKCS : min:256 max:4096 flags:0xA01 Returned: 0 CKR_OK

13: C_OpenSession
[in] slotID = 0x0
[in] flags = 0x6
pApplication=0754D000
Notify=6C5DA378
[out] *phSession = 0x2
Returned: 0 CKR_OK

14: C_GenerateKeyPair
[in] hSession = 0x2
pMechanism->type=CKM_RSA_PKCS_KEY_PAIR_GEN
[in] pPublicKeyTemplate[8]:
    CKA_MODULUS_BITS [size : 0x4 (4)]
    00040000
    CKA_PUBLIC_EXPONENT [size : 0x3 (3)]
    010001
    CKA_TOKEN True
    CKA_DERIVE False
    CKA_WRAP False
    CKA_VERIFY False
    CKA_VERIFY_RECOVER False
    CKA_ENCRYPT False
[in] pPrivateKeyTemplate[7]:
    CKA_TOKEN True
    CKA_PRIVATE True
    CKA_SENSITIVE True
    CKA_DERIVE False
    CKA_UNWRAP False
    CKA_SIGN True
    CKA_DECRYPT True
[out] hPublicKey = 0x4
[out] hPrivateKey = 0x5
Returned: 0 CKR_OK

15: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x4
[in] pTemplate[1]:
    CKA_CLASS requested with 4 buffer
[out] pTemplate[1]:
    CKA_CLASS CKO_PUBLIC_KEY
Returned: 0 CKR_OK

16: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x4
[in] pTemplate[4]:
    CKA_CLASS requested with 4 buffer
    CKA_KEY_TYPE requested with 4 buffer
    CKA_MODULUS requested with 0 buffer
    CKA_PUBLIC_EXPONENT requested with 0 buffer
[out] pTemplate[4]:
    CKA_CLASS CKO_PUBLIC_KEY
    CKA_KEY_TYPE CKK_RSA
    CKA_MODULUS has size 128
    CKA_PUBLIC_EXPONENT has size 3
Returned: 0 CKR_OK
```



```

17: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x4
[in] pTemplate[4]:
    CKA_CLASS            requested with 4 buffer
    CKA_KEY_TYPE          requested with 4 buffer
    CKA_MODULUS           requested with 128 buffer
    CKA_PUBLIC_EXPONENT   requested with 3 buffer
[out] pTemplate[4]:
    CKA_CLASS            CKO_PUBLIC_KEY
    CKA_KEY_TYPE          CKK_RSA
    CKA_MODULUS           [size : 0x80 (128)]
    BA4297A3 B1FEB5B0 B15F0996 23546B4D 4E0496F6 FC5311CD 3051616E 4E697017
    90E79ABB 2F41009E AB0BA92A EF622920 C2E92001 72751918 2B892CB4 246F785E
    C23CDC62 28541021 5591708A 1B539B5B B8578C21 9480E371 9E54A105 9538D353
    2894C651 CCC2EEDA 68E09779 1330D25F 4005146E 5650EFFB E2088911 9B67289B
    CKA_PUBLIC_EXPONENT   [size : 0x3 (3)]
    010001
Returned:  0 CKR_OK

18: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x4
[in] pTemplate[1]:
    CKA_TOKEN            requested with 1 buffer
[out] pTemplate[1]:
    CKA_TOKEN            True
Returned:  0 CKR_OK

19: C_SetAttributeValue
[in] hSession = 0x2
[in] hObject = 0x5
[in] pTemplate[1]:
    CKA_ID                [size : 0x14 (20)]
    FE7732BA 7099D67C 03E0A46D 1024DC0F F650600B
Returned:  0 CKR_OK

20: C_SetAttributeValue
[in] hSession = 0x2
[in] hObject = 0x4
[in] pTemplate[1]:
    CKA_ID                [size : 0x14 (20)]
    FE7732BA 7099D67C 03E0A46D 1024DC0F F650600B
Returned:  0 CKR_OK

21: C_CloseSession
[in] hSession = 0x2
Returned:  0 CKR_OK

22: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x5
[in] pTemplate[1]:
    CKA_PRIVATE           requested with 1 buffer
[out] pTemplate[1]:
    CKA_PRIVATE           True
Returned:  0 CKR_OK

23: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x5
[in] pTemplate[1]:
    CKA_MODULUS           requested with 0 buffer
[out] pTemplate[1]:
    CKA_MODULUS           has size 128
Returned:  0 CKR_OK

```

```

24: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x5
[in] pTemplate[1]:
    CKA_MODULUS                requested with 128 buffer
[out] pTemplate[1]:
    CKA_MODULUS                [size : 0x80 (128)]
    BA4297A3 B1FEB5B0 B15F0996 23546B4D 4E0496F6 FC5311CD 3051616E 4E697017
    90E79ABB 2F41009E AB0BA92A EF622920 C2E92001 72751918 2B892CB4 246F785E
    C23CDC62 28541021 5591708A 1B539B5B B8578C21 9480E371 9E54A105 9538D353
    2894C651 CCC2EEDA 68E09779 1330D25F 4005146E 5650EFFB E2088911 9B67289B
Returned: 0 CKR_OK

25: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x5
[in] pTemplate[1]:
    CKA_PRIVATE                requested with 1 buffer
[out] pTemplate[1]:
    CKA_PRIVATE                True
Returned: 0 CKR_OK

26: C_OpenSession
[in] slotID = 0x0
[in] flags = 0x4
pApplication=0754D000
Notify=6C5DA378
[out] *phSession = 0x2
Returned: 0 CKR_OK

27: C_SignInit
[in] hSession = 0x2
pMechanism->type=CKM_RSA_PKCS
[in] hKey = 0x5
Returned: 0 CKR_OK

28: C_Sign
[in] hSession = 0x2
[in] pData[ulDataLen] [size : 0x22 (34)]
    3020300C 06082A86 4886F70D 02050500 0410D69F EC8728E9 B6BE903F 103FA799
    A07F
[out] pSignature[*pSigLen] [size : 0x80 (128)]
    08D536C7 01BFC89C E27F4F74 DE19DB41 068E73CB 9F34D4A5 B1AF77F7 68FFD6E0
    3005C2FA 42678AC6 EEAC72C4 A1B9216D C383A1A4 5A987014 A51C822F A80D3BD2
    001C204F 091B3877 A498FB42 9D7117FB 4E485F99 4A6C16EE 2DCA7D8C B3D3668F
    9414387B 66F21474 B99AFAB1 5DB6FA30 AEC26320 265DB195 4A253F4A AA226C3F
Returned: 0 CKR_OK

29: C_CloseSession
[in] hSession = 0x2
Returned: 0 CKR_OK

30: C_GetAttributeValue
[in] hSession = 0x1
[in] hObject = 0x4
[in] pTemplate[1]:
    CKA_TOKEN                  requested with 1 buffer
[out] pTemplate[1]:
    CKA_TOKEN                  True
Returned: 0 CKR_OK

```