

# **Knihovny bezpečnostních funkcí Java SE**

## **Java SE Security Libraries**

Pavel Bobek

---

Bakalářská práce  
2015



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2014/2015

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Pavel Bobek**  
Osobní číslo: **A12549**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **prezenční**

Téma práce: **Knihovny bezpečnostních funkcí Java SE**  
Téma anglicky: **Java SE Security Libraries**

Zásady pro vypracování:

1. Prostudujte knihovny v balíčku `java.security` a jejich použití zejména pro práci s certifikáty `X.509`, zabezpečenými úložišti (včetně `smartcards`) a pro podpisování a šifrování dat.
  2. Analyzujte specifika nejrozšířenějších platforem (Win/Lin/Mac) v oblasti přístupu k systémovým a externím úložištím certifikátů (vč. privátních klíčů), popř. ke speciálním zařízením pro šifrování.
  3. Prostudujte způsob podpisování aplikací/appletů. Najděte cenově výhodné kořenové certifikační autority, instalované v běžných prohlížečích a v prostředí JRE/JDK.
  4. Navrhněte a implementujte ukázkovou aplikaci – applet pro podpisování datových souborů, která bude demonstrovat výše uvedené technologie.
  5. Aplikaci otestujte na všech platformách.
-

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. PECINOVSKÝ, Rudolf. Java 8: úvod do objektové architektury pro mírně pokročilé. 1. vyd. Praha: Grada, 2014, 655 s. ISBN 978-80-247-4638-8.
2. BLOCH, Joshua. Java efektivně: 57 zásad softwarového experta. 1. vyd. Praha: Grada Publishing, 2002. ISBN 80-247-0416-1.
3. TILBORG, Henk C. Fundamentals of cryptology: a professional reference and interactive tutorial. Boston: Kluwer Academic Publishers, c2000, xiv, 490 s. ISBN 0792386752.
4. BUDIŠ, Petr. Elektronický podpis a jeho aplikace v praxi: certifikáty a certifikační autority : legislativní rámec elektronického podpisu : praktické aplikace. 1. vyd. Olomouc: ANAG, 2008, 157 s. ISBN 978-80-7263-465-1.
5. MLÝNEK, Jaroslav. Zabezpečení obchodních informací. 1. vyd. Brno: Computer Press, c2007. ISBN 978-80-251-1511-4.
6. Java Platform, Standard Edition 8 API Specification [online]. [cit. 2015-02-05]. Dostupné z: <http://docs.oracle.com/javase/8/docs/api/>

Vedoucí bakalářské práce:

**Ing. Tomáš Dulík, Ph.D.**

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

**6. března 2015**

Termín odevzdání bakalářské práce:

**22. května 2015**

Ve Zlíně dne 6. března 2015



L.S.

doc. Mgr. Milan Adámek, Ph.D.  
*děkan*

prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*


**Prohlašuji, že**

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

**Prohlašuji,**

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

Ve Zlíně

  
.....  
podpis diplomanta

## **ABSTRAKT**

Práce se zabývá studiem knihoven z balíčku java.security. Zaměřuje se na přístup těchto knihoven k úložištím certifikátů X.509. Popisuje přístup těchto knihoven k systémovým úložištím na nejrozšířenějších operačních systémech. Nastiňuje postup při podepisování aplikací. Součástí práce je naprogramovaná aplikace typu applet, která demonstruje prostudované technologie.

Klíčová slova: kryptografie, asymetrická kryptografie, certifikát, certifikační autorita, veřejný klíč, soukromý klíč, Java, applet, podepisování kódu.

## **ABSTRACT**

This work deals with the study of libraries of Java Security package. It targets accessing of storage that holds X.509 certificates. It describes access of this libraries to storage of the most used operating systems. It gives view to procedure of code signing. This work contains applet application that demonstrating this technologies.

Keywords: cryptography, public-key cryptography, certificate, certificate authority, public key, private key, Java, applet, code signing

Za poskytnutí materiálu, věnování času a věcné připomínky bych chtěl poděkovat svému vedoucímu práce Ing. Tomáši Dulíkovi Ph.D.

Pavel Bobek

# OBSAH

<b>ABSTRAKT</b> .....	<b>4</b>
<b>ABSTRACT</b> .....	<b>4</b>
<b>ÚVOD</b> .....	<b>10</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>11</b>
<b>1 KRYPTOGRAFIE</b> .....	<b>12</b>
1.1 SYMETRICKÁ KRYPTOGRAFIE.....	12
1.2 ASYMETRICKÁ KRYPTOGRAFIE.....	12
1.3 CERTIFIKÁTY.....	13
1.4 STRUKTURA CERTIFIKÁTU.....	14
1.4.1 Položka Version.....	14
1.4.2 Položka Serial Number.....	15
1.4.3 Položka Signature Algorithm.....	15
1.4.4 Položka Issuer a Subject.....	15
1.4.5 Položka Validity.....	15
1.4.6 Položka Public Key Info.....	15
1.4.7 Položka Extensions.....	15
1.4.8 Položka Signature.....	16
<b>2 PROGRAMOVACÍ JAZYK JAVA</b> .....	<b>16</b>
2.1 CHARAKTERIZACE JAZYKA JAVA.....	16
2.1.1 Jednoduchý.....	16
2.1.2 Objektově orientovaný.....	17
2.1.3 Distribuovaný.....	17
2.1.4 Interpretovaný.....	17
2.1.5 Robustní.....	17
2.1.6 Bezpečný.....	17
2.1.7 Nezávislý na architektuře.....	18
2.1.8 Přenosný.....	18
2.1.9 Vysoce výkonný.....	18
2.1.10 Víceprocesní.....	18
2.1.11 Dynamický.....	19
2.2 JAVA DNES.....	19
2.3 BALÍČEK SECURITY.....	19
<b>3 ÚLOŽIŠTĚ CERTIFIKÁTŮ</b> .....	<b>19</b>
3.1 SYSTÉMOVÁ ÚLOŽIŠTĚ.....	19
3.1.1 Správa certifikátu na systémech Windows.....	20
3.1.2 Správa certifikátu na systémech OS X.....	20
3.1.3 Správa certifikátů na operačních systémech typu Linux.....	20
3.2 SOUBOROVÁ ÚLOŽIŠTĚ.....	21
3.3 EXTERNÍ ÚLOŽIŠTĚ.....	21
3.3.1 Čipové karty.....	21

<b>4</b>	<b>PODEPISOVÁNÍ APPLETŮ</b>	<b>23</b>
4.1	CODE SIGNING CERTIFIKÁTY	24
4.1.1	Self Signed certifikáty	24
<b>II</b>	<b>PRAKTICKÁ ČÁST</b>	<b>25</b>
<b>5</b>	<b>KNIHOVNA SIGNLIB</b>	<b>26</b>
5.1	TŘÍDA SIGNSERVICEIMPL	27
5.1.1	Metoda getCertTokens	27
5.1.2	Metoda sign	27
5.1.3	Metoda loadCertFile	27
5.1.4	Metoda loadCertCard	27
5.1.5	Metoda loadTerminal	28
5.1.6	Metoda loadCard	28
5.1.7	Metoda isTerminalConnected	28
5.1.8	Metoda isCardPresent	28
5.1.9	Metoda setUserPkcs11Wrapper	28
5.1.10	Metoda initSmartCardWatchdogs	29
5.1.11	Metoda stopSignServiceWatchdogs	29
5.1.12	Metoda eraseSmartCardKeyStore	29
5.1.13	Metoda getPkcs11WrapperList	29
5.1.14	Metoda pkcs11Wrapper2StringMap	30
5.2	ROZHRANÍ KEYSTORESERVICE	30
5.2.1	Metoda getCertTokens	30
5.2.2	Metoda sign	30
5.3	TŘÍDA X509TOKEN	30
5.3.1	Metoda getAlias	30
5.3.2	Metoda getCertificate	31
5.3.3	Metoda getCertificateChain	31
5.3.4	Metoda getId	31
5.3.5	Metoda getPrivateKey	31
5.3.6	Metoda getSource	31
5.3.7	Metoda equals	31
5.3.8	Metoda hashCode	32
5.3.9	Metoda isValidToken	32
5.3.10	Metoda createCertToken	32
5.4	TŘÍDA X509TOKENINTERPRETER	33
5.4.1	Metoda parsePrincipal	33
5.4.2	Metoda getCertToken	33
5.4.3	Metoda interpretCertSubjectName	33
5.4.4	Metoda interpretCASubjectNames	33
5.4.5	Metoda toString	33
5.5	TŘÍDA SYSTEMKEYSTORESERVICEIMPL	34
5.6	TŘÍDA FILEKEYSTORESERVICEIMPL	34
5.6.1	Metoda loadCertFile	34
5.7	TŘÍDA SMARTCARDKEYSTORESERVICEIMPL	35

5.7.1	Metoda setUserPkcs11Wrapper.....	35
5.7.2	Metoda LoadCertCard.....	35
5.7.3	Metoda loadTerminal.....	35
5.7.4	Metoda loadCard.....	36
5.7.5	Metoda isTerminalConnected.....	36
5.7.6	Metoda isCardPresent.....	36
5.7.7	Metoda initSmartCardWatchdogs.....	36
5.7.8	Metoda stopSmartCardWatchdogs.....	36
5.7.9	Metoda eraseKeyStore.....	36
5.7.10	Metoda getPkcs11WrapperList.....	37
5.7.11	Metoda pkcs11Wrapper2StringMap.....	37
5.8	ROZHRANÍ PKCS11WRAPPER.....	37
5.8.1	Metoda getLibratyPath.....	37
5.8.2	Metoda installProvider.....	37
5.8.3	Metoda isValid.....	38
5.8.4	Metoda setEnabled.....	38
5.8.5	Metoda isEnabled.....	38
5.8.6	Metoda uninstallProvider.....	38
5.9	TŘÍDA CZECHEIDWRAPPER.....	38
5.10	TŘÍDA OPENSCTRWRAPPER.....	38
5.11	TŘÍDA USERPKCS11WRAPPER.....	39
5.12	TŘÍDA PKCS11WRAPPERFACTORYIMPL.....	39
5.12.1	Metoda getOpenSCWrapper.....	39
5.12.2	Metoda getCzechEidWrapper.....	39
5.12.3	Metoda getUserPkcs11Wrapper.....	39
5.12.4	Metoda pkcs11Wrapper2StringMap.....	39
5.13	TŘÍDA CARDWATCHDOGEVENTIMPL.....	40
5.13.1	Metoda getEventCardIn.....	40
5.13.2	Metoda getEventCardOut.....	40
5.13.3	Metoda getEvent.....	40
5.13.4	Metoda setEvent.....	40
5.14	ROZHRANÍ CARDWATCHDOGHANDLER.....	40
5.14.1	Metoda handleCardWatchdogEvent.....	40
5.15	ROZHRANÍ CARDWATCHDOG.....	41
5.15.1	Metoda addCardWatchdogHandler.....	41
5.15.2	Metoda setTerminal.....	41
5.15.3	Metoda start.....	41
5.15.4	Metoda stop.....	41
5.15.5	Metoda fireWatchDogEvent.....	41
5.15.6	Metoda isRunning.....	41
5.16	TŘÍDA CARDINWATCHDOG.....	41
5.17	TŘÍDA CARDOUTWATCHDOG.....	42
5.18	TŘÍDA SIMPLETERMINALWATCHDOG.....	42

5.19	BALÍČEK CZ.UTB.SIGNLIB.DERIVATIONS.....	42
<b>6</b>	<b>VÝSTUPNÍ APLIKACE.....</b>	<b>42</b>
6.1	HLAVNÍ OKNO APLIKACE.....	43
6.2	MENU MOŽNOSTI.....	44
6.2.1	Volba jazyka.....	44
6.2.2	Volba Pkcs11 knihovny.....	44
6.2.3	Volba Načti certifikát.....	45
6.2.4	Volba Načti pkcs11 knihovnu.....	45
6.2.5	Volba Načti čipovou kartu.....	45
6.2.6	Volba Používat datové úložiště.....	46
6.2.7	Volba Vyčistit datové úložiště.....	46
6.2.8	Volba Automatická detekce karet?.....	46
<b>7</b>	<b>TESTOVÁNÍ APLIKACE.....</b>	<b>46</b>
7.1	TESTOVÁNÍ NA OPERAČNÍM SYSTÉMU WINDOWS.....	47
7.2	TESTOVÁNÍ NA OPERAČNÍM SYSTÉMU TYPU LINUX.....	47
7.3	TESTOVÁNÍ NA OPERAČNÍM SYSTÉMU OS X.....	48
	<b>ZÁVĚR.....</b>	<b>50</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>51</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>54</b>
	<b>SEZNAM OBRÁZKŮ.....</b>	<b>55</b>
	<b>SEZNAM TABULEK.....</b>	<b>56</b>
	<b>SEZNAM PŘÍLOH.....</b>	<b>57</b>

## ÚVOD

Elektronizace přináší do našeho života mnoho nového. Některé věci, známé a užívané v každodenním životě, získávají i svou elektronickou podobu. Lze mezi ně zařadit i elektronický podpis, ten lze chápat jako obdobu vlastnoručního podpisu. Proto, aby bylo možné tyto technologie úspěšně a beze strachu využívat, je dobré se o nich dozvědět více. [1]

Tato práce má za úkol popsat základní funkcionalitu knihoven z balíčku `java.security` pro přístup k úložištím certifikátů a způsob využití těchto certifikátů k podpisu datových souborů. Práce také demonstruje, jak se liší tento postup na nejrozšířenějších operačních systémech.

V teoretické části této práce shrnuji kryptografii, jako vědní obor, a uvádím její rozdělení na symetrickou a asymetrickou kryptografii. Definuji zde pojem certifikátu, uvádím a rozebírám jeho strukturu. Následně uvádím základní specifikaci jazyka Java. Poté specifikuji jednotlivá úložiště certifikátů. Uvádím, jakým způsobem přistupují knihovny z balíčku `java.security` k jednotlivým úložištím. Nakonec představuji, jakým způsobem je možné podepsat applet naprogramovaný v Javě.

V praktické části popisuji ukázkovou aplikaci pro podepisování datových souborů, kterou jsem naprogramoval jako demonstraci výše zmíněných technologií. Nejprve zde uvádím popis knihovny, která využívá funkcí balíčku `java.security`. Následně popisuji funkci appletu, který využívá tuto knihovnu k podepisování datových souborů. V poslední kapitole uvádím, jakým způsobem probíhal postup testování aplikace na nejčastěji se vyskytujících platformách.

## **I. TEORETICKÁ ČÁST**

## 1 KRYPTOGRAFIE

Kryptografie je dobře zavedený vědní obor, který má na lidskou historii nemalý vliv již více než dva tisíce let. Mezi její nejčastější uživatele patří různé vlády a armádní složky. Před sedmdesátými lety minulého století byla kryptografie jakýmsi tajným uměním, kterému se věnovalo pouze pár jedinců z vládních a armádních kruhů. Nyní představuje zavedený vědní obor vyučovaný na mnoha univerzitách. Nachází široké uplatnění jak v podnikatelském sektoru, tak i pro osobní využití. [2]

### 1.1 Symetrická kryptografie

Metody symetrické kryptografie se vyznačují použitím jediného šifrovacího klíče. Znamená to, že stejný klíč, který byl použit k zašifrování zprávy na straně odesilatele bude použit i na straně příjemce pro dešifrování zprávy. Tento klíč je společným tajemstvím komunikujících stran. Z toho vyplývá nutnost před začátkem komunikace předat důvěryhodným kanálem šifrovací klíč.

### 1.2 Asymetrická kryptografie

V asymetrické kryptografii se oproti kryptografii symetrické užívá dvojice klíčů. Tuto dvojici klíčů si vygeneruje uživatel pomocí některého z běžně dostupných softwarových produktů. Dvojice klíčů bývá označována jako párová data. Pokud lze jeden z klíčů (veřejný klíč) zveřejnit, aniž by bylo možné z něj v reálném čase odvodit druhý klíč (soukromý klíč), označuje se taková kryptografie jako kryptografie s veřejným klíčem.

Princip spočívá v tom, že data šifrovaná jedním z klíčů lze v rozumném čase dešifrovat pouze ze znalosti druhého z dvojice klíčů a naopak. Jeden z nich, tzv. soukromý klíč (někdy se nazývá i privátní) je s maximální únosnou mírou chráněn (ukrývá) majitelem (čipové karty, tokeny, disketa v trezoru atd.), zatímco druhý klíč je naopak zveřejněn (tzv. veřejný klíč).

Byla-li zpráva šifrována soukromým klíčem a příjemce zprávy má k dispozici odpovídající veřejný klíč (a zná jeho vlastníka), kterým zprávu lze dešifrovat, zná odesilatele. Protože je veřejný klíč obecně znám všem, nelze soukromým klíčem šifrovanou zprávu považovat za zašifrovanou v plném smyslu slova (důvěrnou), ale pouze za autorizovanou (nepopíratelnou). Což je principem elektronického podpisu.

### 1.3 Certifikáty

Řešením problému správy, distribuce a uchování klíčů je využití takzvaného certifikátu veřejného klíče, zkráceně nazývaného certifikát. Certifikát lze z jistého úhlu pohledu chápat jako obdobu průkazu totožnosti, například občanského průkazu.

Certifikáty obsahují ve své nejjednodušší formě veřejný klíč, jméno a další údaje, zajišťující jednoznačnou identifikaci subjektu, kterému byl tento certifikát vydán. Běžně používané certifikáty též obsahují datum počátku platnosti, jméno certifikační autority, která certifikát vydala, sériové číslo a některé další informace.

Vydavatelem certifikátu může být kdokoliv, kdo má k dispozici příslušnou technologii, zpravidla to jsou však specializovaní poskytovatelé certifikačních služeb (PCS). Častěji je v praxi pro tyto instituce užíván název certifikační autorita (CA). Certifikační autorita vystupuje při vzájemné komunikaci dvou subjektů jako třetí, nezávislý a důvěryhodný subjekt, který prostřednictvím jím vydaného certifikátu jednoznačně svazuje identifikaci subjektu (fyzickou identitu) s jeho dvojicí klíčů (elektronickou identitou), respektive následně s jeho elektronickým podpisem. Certifikační autorita garantuje jedinečnost subjektů podle užití identifikace subjektů v rámci vydavatelských certifikátů. To je zajištěno legislativními a technickými pravidly provozu instituce CA. [1, s. 28-40]

### 1.4 Struktura certifikátu

Struktura certifikátů používaných pro komunikaci prostřednictvím internetu je popsána v doporučení RFC-3280 (Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile). Tento dokument specifikuje jednotlivé části certifikátu a udává jeho formu. Strukturu certifikátu je možné vidět na obrázku (Obr. 1).

```

[
[
Version: V3
Subject: CN=Pavel Bobek, OU=FAI, OU=UTB, L=Zlín, ST=Czech Republic, C=CZ
Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11
Key: Sun RSA public key, 2048 bits
modulus:
279507875662233468736239493601968471333965619839622639320735665224
538323659614717011907303595694128793420974522463238656073268708849
469143532186634259708649414908244782990477309358753936888076674905
706372203007323315123449222238376202421462395595341659839389193204
548803815802882957243000202353817916019100308307562069520407562785
628390065058013725559075346550593456334923484487488083134077852809
888442111728538936330964221753983872644026378712123849706975504177
806592837318197020888170512059847514765389329011349588276600825482
674251415635078401610579475392281919363698606596300780762133814345
78603145465880239277087
public exponent: 65537
Validity: [From: Wed Mar 25 15:15:32 CET 2015,
To: Sat Mar 24 15:13:32 CET 2018]
Issuer: CN=Pavel Bobek, OU=FAI, O=UTB, L=Zlín, ST=Czech Republic, C=CZ
SerialNumber: [ 376493b4]
Certificate Extensions: 1
[1]: ObjectID: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 1C 1A 55 3B 9A 23 E5 20 51 9A D7 FE 09 85 9D CB ...U;#. Q.....
0010: 8D E1 82 F8 .....
]
]
Algorithm: [SHA256withRSA]
Signature:
0000: 43 73 08 01 5A DA 5F C4 E9 1D D1 1F EE A6 F6 4B Cc..Z.....K
0010: 3B 3F E5 07 85 27 F5 0F 8D 62 25 AB 20 8E A2 58 ;?.....bN. .X
0020: 2F 8E 73 E0 2B 38 A6 A5 54 81 E1 EA 84 97 97 2D /..s.+B..T.....-
0030: 34 9D 47 F8 68 4C FD FC 94 51 C2 08 0D 12 8D 87 4.G.hL...Q.....
0040: D1 A1 2C 72 4F 61 59 A6 8D D0 5B 46 52 48 11 FF ...rOaY...[F.K...
0050: 3F CA 54 10 86 C3 37 D2 E8 8D 65 C 00 03 A3 53 ?.T...7..mk...S
0060: 2B 84 27 14 04 9C 35 F1 76 9B 7 3C 0B FE DE 5E +...5.v..f.....^
0070: D1 D9 34 1B E3 75 89 F5 59 FA 7 DC CB 52 75 98 ...4..u..Y...Ru..
0080: 21 25 AB 88 34 1F 84 2C 2E DC 57 9B 42 AC 86 D1 !%..4...W..B...
0090: 1E 5E ED C4 52 98 1E 3A A4 76 C2 F6 9B FB 28 1D ..^..R.....v...(.
00A0: 10 68 4F AA A0 13 83 8E 64 31 46 74 3C 92 30 D7 .kO.....d1P<.0.
00B0: EE 5A 8C 34 EB F4 2A E7 11 9E 1F CE F9 F2 63 50 ..2.4..*.....cP
00C0: 8B 05 66 DA 44 AF 95 FC 81 0E B4 90 64 30 01 E3 ..f.D.....dB...
00D0: 53 C5 1C 5F DE CE E3 83 98 9D C5 28 60 7F 5F 16 S.....+.....^
00E0: F7 ED D3 61 1C 3A CE 36 76 A0 8B 32 A4 15 25 D8 ...a..6...2..%.
00F0: 8B 84 5F 30 C1 3C 39 95 61 8A 11 42 A7 C8 F6 43 ..._0.<9..a..B...C
]
]
    
```

Obr. 1: Struktura certifikátu X.509v3

#### 1.4.1 Položka Version

Tato část certifikátu udává o jakou verzi se jedná. V současné době je využíváným certifikátem X.509 verze 3.

#### **1.4.2 Položka Serial Number**

V této položce je uložen jednoznačný identifikátor certifikátu. Tento identifikátor musí být kladné celé číslo. Dále musí být unikátně, v rámci jedné certifikační autority, přiřazován svým certifikátům.

#### **1.4.3 Položka Signature Algorithm**

V této položce je uvedeno, který algoritmus byl využit k podpisu tohoto certifikátu jeho certifikační autoritou. Znalost tohoto algoritmu je nutná pro zpětné ověření platnosti certifikátu.

#### **1.4.4 Položka Issuer a Subject**

Tato položka obsahuje informace, které identifikují poskytovatele popřípadě vlastníka certifikátu. Tato položka musí obsahovat neprázdný seznam hodnot (Distinguished name - DN) o poskytovateli.

#### **1.4.5 Položka Validity**

Zde jsou uloženy dva časové údaje, které udávají dobu po kterou je certifikát platný. Údaj označovaný jako počátek platnosti (notBefore) udává dobu od které je certifikát platný. Dobu vypršení platnosti certifikátu udává hodnota ukončení doby platnosti (notAfter).

#### **1.4.6 Položka Public Key Info**

Tato položka obsahuje veškerá data spojená s veřejným klíčem vlastníka certifikátu. Je zde uveden algoritmus, který je použit pro vytvoření klíče. Dále jsou zde uvedeny vlastní hodnoty klíče.

#### **1.4.7 Položka Extensions**

X.509 verze 3 se od svých předchozích verzí liší hlavně přidáním extenzí. Tyto rozšíření umožňují přidat certifikátu další atributy, které se mohou týkat jeho vlastníka, popřípadě klíčů, nebo mohou sloužit pro správu hierarchie certifikátů.

### 1.4.8 Položka Signature

Poslední položkou certifikátu je jeho podpis certifikační autoritou. Zde je uložen podpis certifikační autority tohoto certifikátu. [3]

## 2 PROGRAMOVACÍ JAZYK JAVA

Vývoj programovacích jazyků je řízen dvěma faktory – vývojem v metodice programování a vývojem operačních systémů. Programovací jazyk Java není výjimkou. Je založen na dědictví programovacích jazyků C a C++, navíc obsahuje další vlastnosti odpovídající požadavkům na vývoj moderních aplikací. Vývoj Javy reagoval na růst prostředí „on-line“ a také na rostoucí požadavky distribuovaných počítačových systémů.

Základní koncepci tomuto jazyku definovali pánové James Gosling, Patrik Naughton, Cris Warth, Ed Frank a Mike Sheridan. Všichni byli zaměstnanci firmy Sun Microsystems. Jejich programovací jazyk byl původně označován jako „Oak“. V roce 1995 byl přejmenován na „Java“.

Hybným momentem pro vznik jazyka Java byla potřeba programovacího jazyka nezávislého na platformě. Většina programovacích jazyků existujících v době vzniku jazyka Java byla navržena tak, že bylo možné překládat jejich aplikace pouze pro jisté počítačové platformy. [4]

### 2.1 Charakterizace jazyka Java

Javu lze popsat jako jednoduchý, objektově orientovaný, distribuovaný, interpretovaný, robustní, bezpečný, nezávislý na architektuře, přenosný, vysoce výkonný, víceprocesní a dynamický jazyk. V současné době se jedná o jeden z nejoblíbenějších programovacích jazyků a s jeho aplikací se můžeme setkat na mnoha místech, od čipových karet, přes aplikace pro webové prostředí, aplikace pro stolní počítače i aplikace pro mobilní telefony.

#### 2.1.1 Jednoduchý

Jedním z cílů návrhářů bylo vytvořit jazyk, který se může programátor naučit velmi rychle. Je snaha udržovat počet jazykových konstrukcí na nízké úrovni. Dalším cílem návrhu jazyka je, aby byl snadno programátory uchopitelný. Syntaxe jazyka Java je velmi podobná jazyku C nebo C++. Java používá řadu stejných jazykových konstrukcí jako C a C++.

### 2.1.2 Objektově orientovaný

Java je objektově orientovaný jazyk. V objektově orientovaném systému je třída (class) souborem dat a metod, které operují nad těmito daty. Společně tak data a metody popisují stav a chování objektu. Třídy jsou uspořádány do hierarchie, takže podtřída může zdědit chování své nadtříd. Java přichází se širokou množinou tříd, uspořádaných do balíků (packages), které můžeme ve svých programech využít.

### 2.1.3 Distribuovaný

Java je navržena pro podporu aplikací na sítích. Podporuje různé úrovně síťového spojení prostřednictvím tříd v balíku java.net. Třída URL tak například umožňuje aplikaci v Javě otevřít vzdálené objekty na Internetu a přistupovat k nim. S Javou je tak otevření vzdáleného souboru stejně snadné, jako by to byl soubor lokální. Java taktéž podporuje spolehlivá síťová spojení pomocí kanálů v třídě Socket, takže můžeme vytvářet distribuované klienty a servery.

### 2.1.4 Interpretovaný

Kompilátor Javy vytváří místo skutečného strojového kódu tzv. Bajtové kódy. Abyste program v Javě skutečně spustili, musíte pomocí interpretu Javy přeložené bajtové kódy vykonat (provést), takže Java je interpretovaný jazyk. Program v Javě je možné spustit na kterémkoliv systému, na kterém běží interpret Javy a systém run-time. Dohromady interpret a systém run-time vytváří virtuální stroj, nazývaný Virtuální stroj Javy (Java Virtual Machine).

### 2.1.5 Robustní

Java byla navržena pro psaní vysoce robustního softwaru. Java je silně typovaný jazyk, díky čemuž je možné ve značné míře odhalit potenciální problémy v nesprávném typu již v době kompilace. Java vyžaduje explicitní deklaraci metod. Nepodporuje implicitní deklarace, což zajišťuje, že již kompilátor může odchytit chyby ve vyvolání metod, což vede ke spolehlivějším programům.

### 2.1.6 Bezpečný

Jeden z nejvíce ceněných aspektů Javy. Bezpečnost je velmi důležitou otázkou, protože Java je určena k používání v síťovém prostředí. Řada obraných mechanismů v Javě je urč-

na právě pro ochranu před nedůvěryhodnými applety. Run-time systém Javy pomocí procesu verifikace bajtového kódu zajistí, že kód načtený přes síť nenarušuje žádná z jazykových omezení Javy.

### 2.1.7 Nezávislý na architektuře

Programy v Javě se překládají do formátu bajtového kódu, nezávislého na architektuře. Primární výhodou tohoto přístupu je, že umožňuje spuštění aplikace v Javě na kterémkoliv systému, pokud je na tomto systému implementován virtuální stroj Javy (Java Virtual Machine). To umožňuje vytvářet aplikace, které běží na systémech typu Windows, Linux a iOS.

### 2.1.8 Přenosný

Nezávislost na architektuře je velkou částí přenositelnosti. Java však jde v tomto směru ještě dále. Zajišťuje, že neexistují žádné „implementačně závislé“ aspekty jazyka. Java tak například explicitně určuje velikost každého z primitivních datových typů, stejně jako jejich chování v aritmetických operacích.

### 2.1.9 Vysoce výkonný

Java je interpretovaný jazyk, takže nikdy nebude stejně rychlý jako kompilovaný jazyk, jakým je například C. Ve skutečnosti je Java v průměru 20-krát pomalejší než C. Pro většinu aplikací je ovšem tato rychlost bohatě postačující. V případě kdy je výkon kritický, poskytuje JVM překladače typu „právě včas“ („just in time“), které v době běhu přeloží bajtové kódy do strojového kódu pro konkrétní procesor. Výkon bajtového kódu převedeného do strojového kódu je srovnatelný s jazykem C.

### 2.1.10 Víceprocesní

V síťové aplikaci, založené na grafickém uživatelském rozhraní (GUI) a běžící ve webovém prohlížeči, si je snadné představit, že probíhá několik věcí současně. Uživatel tak může poslouchat zvukový záznam, zatímco prochází stránkou a v pozadí prohlížeč právě natahuje obrázek. Java je víceprocesní (multithreaded) jazyk. Poskytuje podporu pro více prováděcích vláken (kterým se říká lehké procesy), které mohou vykonávat různé úkoly.

### 2.1.11 Dynamický

Jazyk Java byl navržen tak, aby se dokázal přizpůsobit vyvíjejícímu se prostředí. Java tak například načítá (zavádí) třídy podle potřeby, a to i po síti. Třídy v Javě mají také svoji reprezentaci v době běhu (run-time). Run-time definice tříd v Javě umožňují dynamicky linkovat třídy do běžících systémů. [5]

## 2.2 Java dnes

Za posledních pár let prošla Java mnoha změnami. Momentálně je dnes v provozu verze Java 8. Jejím vlastníkem je firma Oracle. Tato verze poskytuje spoustu bezpečnostních záplat. Nové možnosti programování jako jsou lambda výrazy nebo virtuální rozšiřovací metody. [6]

## 2.3 Balíček Security

Technologie Java security poskytuje rozsáhlé aplikační rozhraní, nástroje, implementaci běžně používaných bezpečnostních algoritmů, mechanismů a protokolů. Její aplikační rozhraní má širokou oblast záběru. Zaměřuje se především na kryptografii, infrastrukturu veřejného klíče, bezpečnou komunikaci, autentizaci a kontrolu přístupu. Poskytuje vývojářům obsáhlý bezpečnostní rámec pro psaní aplikací. Uživatelům a správcům poskytuje nástroje pro bezpečnou správu aplikací. [7]

## 3 ÚLOŽIŠTĚ CERTIFIKÁTŮ

V této kapitole se budu zabývat místy, která slouží k ukládání certifikátů a jejich správou. Jelikož pro skladování certifikátů existuje značné množství možností, budu se zde zabývat pouze těmi, na které se primárně zaměřuje tato práce. Dále v této kapitole uvedu, jakým způsobem k daným úložištím přistupuje programovací jazyk Java.

### 3.1 Systémová úložiště

Některé systémy poskytují centrální systémové úložiště. Toto úložiště spravuje přístup k záznamům s uloženými certifikáty, uložená hesla uživatele a mnoho dalšího.

### 3.1.1 Správa certifikátu na systémech Windows

Operační systém poskytuje službu pro správu certifikátů. Tato služba umožňuje správu certifikátu (export, import, úpravu, nebo jejich vytvoření). Tato služba se spouští ze souboru nacházejícího se standardně na adrese „C:\Windows\System32\certmgr.msc“. [8]

Tato služba je zpřístupněna implementací poskytovatele kryptografických služeb knihovny Security „SunMSCAPI“ v Javě. Tento poskytovatel dodává rozšíření třídy KeyStore a umožňuje volání metod této třídy nad službou správy certifikátů. [9]

### 3.1.2 Správa certifikátu na systémech OS X

Operační systémy typu OS X poskytují svým uživatelům tzv. „klíčenku“ („keychain“). Tato aplikace umožňuje bezpečné ukládání hesel, klíčů certifikátů a poznámek. [10]

Za tyto služby v Javě zodpovídá poskytovatel kryptografických služeb „Apple“. Ten dodává rozšíření pro třídu KeyStore a umožňuje volání jejích metod nad klíčenkou systému a umožňuje přístup k certifikátům, které jsou v zde uloženy. [11]

### 3.1.3 Správa certifikátů na operačních systémech typu Linux

Počítačové operační systémy typu Linux mají velké množství softwarů pro uchovávání certifikátu. Tyto systémy jsou často velmi rozmanité a různě kvalitní.

Jako příklad můžeme uvést klíčenku (GNOME Keyring), dostupnou na operačním systému Ubuntu. Tato klíčenka slouží k uchovávání hesel a certifikátů v systému a k jejich distribuování k aplikacím. K práci s certifikáty je potřeba doinstalovat u této klíčenky modul NSS modutil. [12]

Fakt, že těchto softwarů je velké množství a jejich přístup k certifikátům a jejich následná distribuce je různá, je nejspíše důvodem, že balíček java.security neposkytuje podporu těmto aplikacím. V tomto balíčku není implementován žádný poskytovatel služeb, který by implementoval systémové úložiště pro systém Linux, jak je tomu u předchozích systémů. [13]

## 3.2 Souborová úložiště

Soubory, které přenášejí certifikáty, musí splňovat PKCS#12 (Personal information exchange syntax standard) standard. Tento standard popisuje syntaxi pro přenos informací o osobní identitě (včetně soukromých klíčů) a certifikátů. Tyto soubory mívají obvykle příponu souboru „.p12“ nebo „.pfx“. [14]

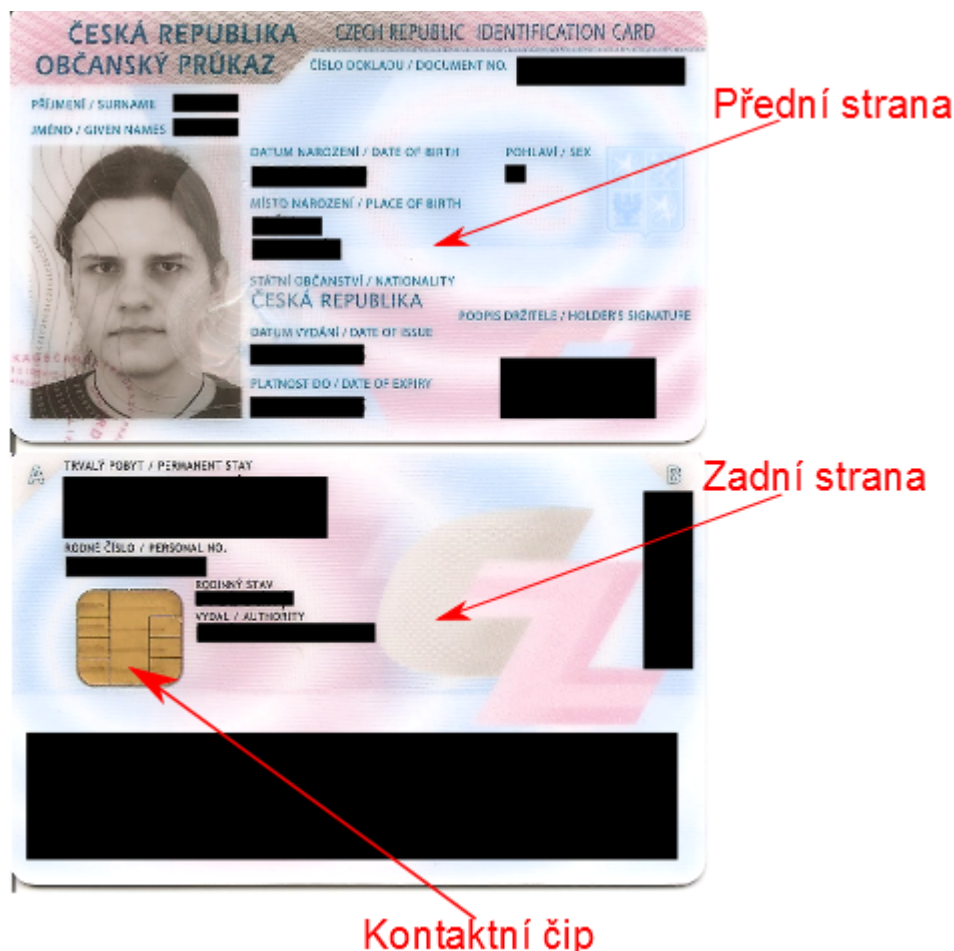
Implementaci tohoto standardu zajišťuje v Javě poskytovatel kryptografických služeb „SunJSSE“. Ten poskytuje implementaci třídy KeyStore která umí pracovat s těmito soubory. [15]

## 3.3 Externí úložiště

Mezi tyto úložiště patří především čipové karty a USB tokeny. Jejich hlavní předností je odnímatelnost, kdy je možné je odebrat a vložit do příslušných čteček či zařízení jen v případě potřeby. Další předností těchto zařízení je možnost jejich konstrukce takovým způsobem, aby je jejich soukromé klíče nemusely (v určitých konstrukčních řešeních ani nemohly) opustit. To zvyšuje zabezpečení těchto klíčů. Čipová karta (nebo USB token) může tedy sloužit jako prostředek pro vytváření elektronického podpisu či jako nosič dat pro vytváření elektronického podpisu. [16]

### 3.3.1 Čipové karty

Kontaktní čipovou kartou je integrovaný obvod, který je zalisovaný v plastovém obalu. Čipová karta je definována standardem ISO 7816 1-6. Tento standard udává fyzickou podobu čipové karty, rozměry a umístění čipu, elektrické charakteristiky a protokol komunikace s kartou, charakteristiku komunikace s kartou, identifikátory poskytnuté kartou a jejich význam a interdisciplinární datové prvky. [17] Detail čipové karty je na obrázku (Obr. 2).

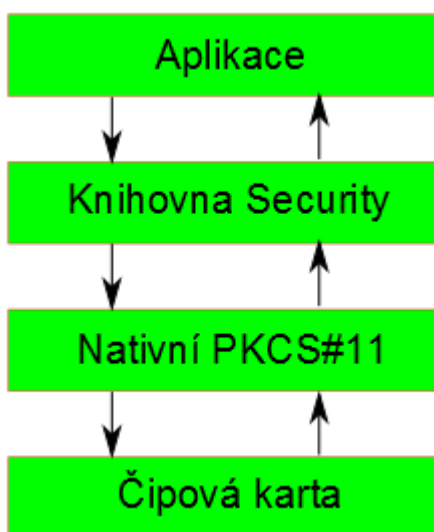


Obr. 2: Občanský průkaz osazený kontaktním čipem

Pro komunikaci čipové karty s aplikací využívá knihovna Security aplikační rozhraní PKCS#11. Poskytovatel kryptografických služeb tohoto rozhraní zde neimplementuje algoritmy. Slouží jako most mezi aplikačním rozhraním knihovny Security a implementací PKCS#11 aplikačního rozhraní pro danou kartu. Ke své činnosti vyžaduje knihovnu, která implementuje aplikační rozhraní PKCS#11 v2.0 nebo vyšší. Poskytovatel potřebuje ke své činnosti konfigurační soubor.

Tento soubor mu je dodán v argumentech konstruktoru. Tento soubor obsahuje konfiguraci pro používání knihovny poskytující přístup k čipové kartě. Je možné jej buďto načíst staticky nebo vytvořit dynamicky za běhu programu. [18]

Princip této komunikace je znázorněn na obrázku (Obr. 3).



Obr. 3: Komunikace aplikace s čipovou kartou

#### 4 PODEPISOVÁNÍ APPLLETŮ

Firma Oracle na svém blogu uvádí, že v lednu roku 2014 vyšla verze Javy 7u51. Od této verze mají programátoři povinnost podepisovat veškeré applety a webové aplikace a přidávat k nim Manifest obsahující definici práv. Pokud tak neučiní, aplikace běžící v prohlížečích, budou zablokovány. [19]

Aplikaci je možno podepsat s využitím RSA certifikátu. Tento certifikát je nutné si nejprve obstarat. Nový certifikát je možné získat využitím aplikace keytool dodávané s distribucí Javy. Po spuštění prvního příkazu z obrázku (Obr. 4) a vyplnění příslušných údajů, je uživateli vygenerován certifikát, který není podepsaný. Aplikací druhého příkazu z obrázku dostane uživatel certifikát ve tvaru žádosti o podpis, ten zadá do webového formuláře dané certifikační autority a obratem dostane soubor, který naimportuje do aplikace keytool třetím příkazem a následně může podepisovat aplikace pomocí aplikace jarsigner, která se nachází ve stejné složce. [20]



Obr. 4: Příkazy pro aplikaci keytool

## 4.1 Code Signing certifikáty

Jedná se o certifikáty, které jsou určeny k podepisování aplikací. Certifikáty umožňují vývojářům a softwarovým firmám podepsat aplikaci určenou k distribuci po internetu. Uživatel aplikace díky podpisu ví, od koho aplikace pochází, a že nebyla během přenosu změněna. Code Signing certifikát je možné pořídit u patřičné certifikační autority. Cena těchto certifikátů se různí. Jedna z nejlepších nabídek je podána firmou K software. Ta udává nabídku při zakoupení certifikátu na pět let cenu \$73 za rok. V přepočtu na české koruny se cena pohybuje okolo 1757kč za rok. [21]

### 4.1.1 Self Signed certifikáty

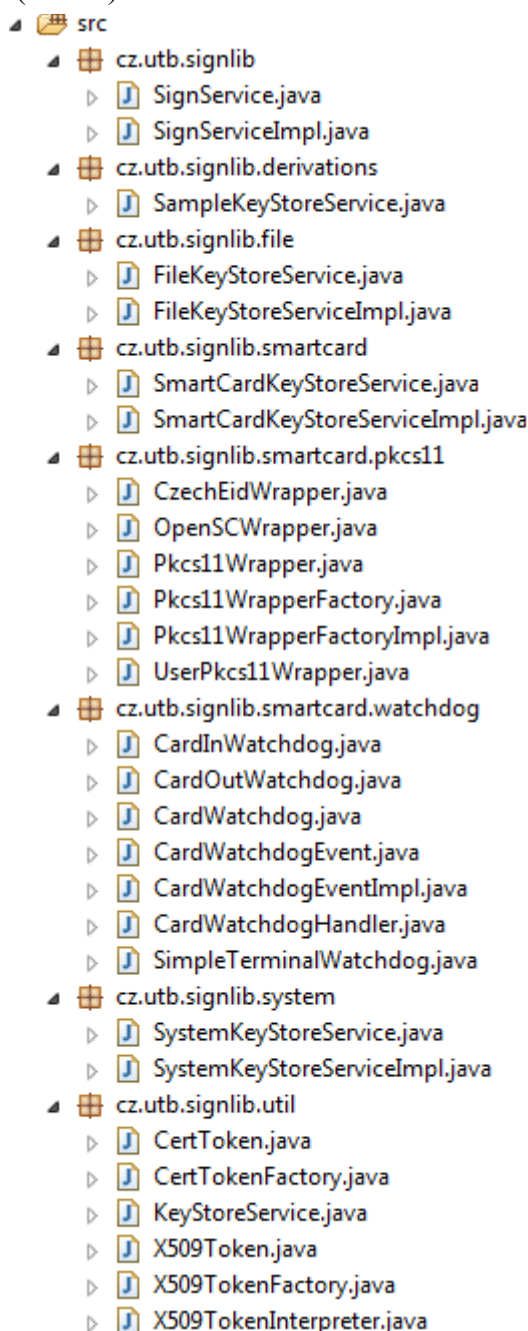
Jedná se o certifikáty, které mají svůj certifikát podepsaný svým soukromým klíčem. Každá kořenová certifikační autorita je tedy tímto certifikátem. Tento certifikát je možné vytvořit například pomocí nástroje keytool nacházejícího se ve složce Javy. Applet podepsaný tímto certifikátem bude blokován. Je možné ale tento certifikát importovat do prostředí Javy na počítači klienta a tím umožnit jeho spuštění. [22]

## **II. PRAKTICKÁ ČÁST**

## 5 KNIHOVNA SIGNLIB

V této kapitole popíši knihovnu SignLib, kterou jsem vytvořil pro práci s certifikáty X509.v3. Tato knihovna umožňuje pracovat s certifikáty uloženými v systémových úložištích, v samostatných souborech a na čipových kartách. Tato knihovna se nachází v samostatném projektu a do appletu je přidána externí závislostí formou jar souboru.

Tato knihovna se skládá z celkem osmi balíčků. V následujících podkapitolách podrobně rozeberu a popíši jednotlivé části této knihovny a jejich funkcionalitu. Struktura knihovny je zobrazena na obrázku (Obr. 5).



Obr. 5: Struktura knihovny SignLib

## 5.1 Třída `SignServiceImpl`

Tato třída zpřístupňuje veškerou funkcionalitu této knihovny. Poskytuje veškeré metody nutné pro práci s certifikáty a jejich úložišti. Třída se nachází v balíčku `cz.utb.signlib`. Třída sestává ze dvou souborů.

Prvním je rozhraní `SignService`. Toto rozhraní shrnuje veškerou funkcionalitu knihovny `SignLib`. Poskytuje deklarace a kontrakty pro metody, které využívají funkcionality této knihovny.

Druhý soubor s názvem `SignServiceImpl`, obsahuje implementaci daného rozhraní formou třídy. Tato třída poskytuje definice k daným metodám, které dále popisují níže v podkapitolách.

### 5.1.1 Metoda `getCertTokens`

Tato metoda ve své návratové hodnotě poskytuje veškeré certifikáty, které se podařilo knihovně načíst. Certifikáty jsou předány formou mapy. Klíčem v této mapě je třída, ze které jsou záznamy pořízeny, a hodnotou je množina záznamů jednotlivých certifikátů.

### 5.1.2 Metoda `sign`

Tato metoda převezme záznam certifikátu a data, která mají být podepsána. Funkcionalita je zde převzata z metody `sign` ve třídě `FileKeyStoreService`.

### 5.1.3 Metoda `loadCertFile`

Tato metoda se pokusí ze souboru, který je jí předán prvním parametrem, načíst všechny certifikáty.

### 5.1.4 Metoda `loadCertCard`

Tato metoda se pokusí načíst čipovou kartu vloženou do čtečky čipových karet.

### 5.1.5 Metoda loadTerminal

Tato metoda načte čtečku čipových karet. Následně nastaví vnitřní chování třídy SmartCardKeyStoreService podle toho, zda-li je k počítači připojena čtečka čipových karet. V tomto případě volá tato metoda stejnojmennou metodu na výše uvedené třídě. Její podrobnější popis se nalézá v dokumentaci dané třídy.

### 5.1.6 Metoda loadCard

Tato metoda načte čipovou kartu vloženou do čtečky čipových karet. Následně nastaví vnitřní chování třídy SmartCardKeyStoreService podle toho, zda-li je do čtečky čipových karet vložena čipová karta. Stejně jako v předchozím případě tato metoda volá stejnojmennou metodu na výše uvedené třídě. Její podrobnější popis je možno nalézt v dokumentaci dané třídy.

### 5.1.7 Metoda isTerminalConnected

Tato metoda vrací ve své návratové hodnotě informaci o tom, jestli je k počítači připojena čtečka čipových karet. Pokud je čtečka čipových karet připojena, vrací tato metoda pravdivostní hodnotu true. V opačném případě je návratová hodnota false. Stejně jako v případech uvedených výše, metoda volá stejnojmennou metodu na třídě SmartCardKeyStoreService. Její podrobnější popis je proto uveden v dokumentaci dané třídy.

### 5.1.8 Metoda isCardPresent

Tato metoda vrací informaci o tom, zda-li je do čtečky čipových karet vložena čipová karta. V případě, že je čipová karta vložena, vrací pravdivostní hodnotu true. V opačném případě je návratová hodnota false. V této třídě vrací metoda pouze výsledek volání na stejnojmenné metodě ve třídě SmartCardKeyStoreService. Její chování bude proto podrobněji popsáno v dokumentaci k této metodě.

### 5.1.9 Metoda setUserPkcs11Wrapper

Tato metoda sestaví ze vstupních parametrů obalovací třídu pro knihovnu, jejíž umístění je dodáno metodě v prvním parametru. Nastaví této třídě alias dodaný v druhém parametru. Pokud není alias poskytnut (hodnota druhého parametru je rovna null), bude mít třída tento parametr automaticky vygenerovaný svou tovární třídou. V této třídě metoda pouze volá

stejnou metodu poskytnutou třídou SmartCardKeyStoreService. Její chování bude proto více rozebráno v dokumentaci k této třídě.

#### **5.1.10 Metoda initSmartCardWatchdogs**

Tato metoda nastaví třídy, které detekují připojení čtečky čipových karet a vložení karty nebo její vyjmutí. Parametrem této metody je třída, která implementuje rozhraní CardWatchdogHandler a tím poskytuje metody pro obsluhu těchto tříd. Funkčnost těchto tříd je popsána v samostatných podkapitolách níže. Tato metoda opět pouze volá stejnojmennou metodu třídy SmartCardKeyStoreService, proto je její chování dále rozvedeno v dokumentaci této třídy.

#### **5.1.11 Metoda stopSignServiceWatchdogs**

Tato metoda ukončí činnost všech tříd, které obstarávají pravidelný dohled nad stavem událostmi, které jsou spojené s čipovou kartou. V momentální implementaci pouze zavolá metodu knihovny SmartCardKeyStoreService k zastavení jejich dozorčích tříd.

#### **5.1.12 Metoda eraseSmartCardKeyStore**

Funkcí této metody je odstranění všech záznamů certifikátů pocházejících z čipových karet. Tato metoda je zde poskytnuta hlavně proto, že její volání je hlavní náplní metod poskytnutých v třídě obsluhujících dozorčí metody třídy SmartCardKeyStoreService. Jedinou funkcí této metody je zavolání metody eraseKeyStore na výše zmíněné třídě.

#### **5.1.13 Metoda getPkcs11WrapperList**

Tato metoda slouží k obdržení všech obalovacích tříd pro knihovny poskytující rozhraní pkcs11. Tyto třídy jsou navraceny v datové kolekci typu list. Její podrobnější chování bude zdokumentováno v podkapitole věnující se třídě SmartCardKeyStoreService, jelikož metoda volá stejnojmennou metodu na této třídě.

#### 5.1.14 Metoda pkcs11Wrapper2StringMap

Tato metoda převezme jako svůj parametr kolekci obalovacích tříd knihoven poskytujících rozhraní pkcs11 typu list. Tyto třídy převede do jednoznačně rozlišitelné textové podoby a vrátí je v kolekci typu mapa. Podrobnější popis je možno najít v dokumentaci třídy SmartCardKeyStoreService. V této třídě totiž metoda volá pouze stejnojmennou metodu uvedené třídy.

### 5.2 Rozhraní KeyStoreService

Toto rozhraní poskytuje základní podklad pro všechny servisní třídy, které přistupují k certifikátům. Je umístěno v balíčku cz.utb.signlib.util.

#### 5.2.1 Metoda getCertTokens

Tato metoda vrací certifikáty dané servisní třídy v kolekci typu list.

#### 5.2.2 Metoda sign

Tato metoda slouží k podepsání dat. Prvním argumentem této metody je instance certifikátové kontejnerové třídy. Druhým argumentem jsou data, která mají být podepsána. Metoda vrací podepsané data. V případě neúspěchu vrací prázdnou hodnotu.

### 5.3 Třída X509Token

Tato třída je navržena pro uchovávání certifikátů v aplikaci. Po vyzvednutí z úložiště je certifikát vložen do této třídy tovární třídou. Tato třída poskytuje základní data certifikátu. Je umístěna v balíčku cz.utb.signlib.util. Je tvořena dvěma soubory. Prvním souborem je rozhraní CertToken. V tomto souboru se nalézají deklarace tříd.

Druhým souborem je X509Token, který implementuje toto rozhraní a poskytuje definici metod uvedených níže.

#### 5.3.1 Metoda getAlias

Tato metoda vrací hodnotu řetězce, pod kterým je uložen daný certifikát v úložišti.

### 5.3.2 Metoda `getCertificate`

Tato metoda ve své návratové hodnotě vrací certifikát, který je na nejnižší úrovni v řetězu certifikátů. Tedy se jedná o uživatelův certifikát. Tento certifikát obsahuje uživatelův veřejný klíč. S tímto klíčem je možné ověřit digitální podpis.

### 5.3.3 Metoda `getCertificateChain`

Tato metoda vrací celý řetězec certifikátů, které obsahuje tato třída.

### 5.3.4 Metoda `getId`

Tato metoda vrací unikátní identifikační hodnotu, která byla instanci přiřazena tovární třídou. Tato hodnota se ovšem může lišit pro stejný certifikát při dvou různých bězích programu.

### 5.3.5 Metoda `getPrivateKey`

Tato metoda vrací soukromý klíč přiřazený k danému certifikátu.

### 5.3.6 Metoda `getSource`

Tato metoda vrací servisní třídu, ve které byla vytvořena tato instance. Hlavním důvodem zde je, že při další práci s certifikáty je možné identifikovat jejich zdroj. Pokud tyto certifikáty pocházejí z nestálého zdroje jako je čipová karta. Tak je možné ověřit jejich platnost.

### 5.3.7 Metoda `equals`

Tuto metodu dědí každá třída v Javě. Jsou zde porovnávány dva objekty. V případě jejich totožnosti je vrácena pravdivostní hodnota `true`. V opačném případě hodnota `false`. Definici této metody obsahuje již základní třída `Object`. Tato metoda je zde přepsána proto, že ji využívají kolekce k uchování instancí. Jelikož jsou instance této třídy předávány v kolekcích, je vhodné, přepsat tuto metodu, aby bylo dosaženo správného chování kolekcí.

Tato metoda zde porovnává dvě instance tím způsobem, že porovná certifikáty na nejnižší úrovni v řetězu certifikátů. Pokud se tyto certifikáty shodují, vrátí metoda návratovou hodnotu `true`. Pokud jsou odlišné, vrátí návratovou hodnotu `false`. Tím docílíme toho, že

dvě instance budou považovány za totožné právě v případě, když budou jejich certifikáty na nejnižší úrovni identické.

### 5.3.8 Metoda hashCode

Tato metoda generuje číselný identifikátor, je definována v nadtřídě Object. Přiřazuje každé instanci třídy číslo. Tuto metodu také využívají kolekce k práci s instancemi. A od kvality jejího napsání se odvíjí efektivita práce kolekcí. [23]

Tato třída přejímá chování stejnojmenné metody zavolané na instanci certifikátu na nejnižší úrovni certifikačního řetězu. Tímto docílíme toho, že instancím této třídy bude přidělován identifikátor tohoto certifikátu.

### 5.3.9 Metoda isValid

Tato metoda kontroluje, zda-li je certifikát v této instanci platný. Pokud instance obsahuje řetěz certifikátů, soukromý klíč a identifikační řetězec, vrací tato metoda hodnotu true. V opačném případě je návratová hodnota false.

Metoda vyhazuje také dvě výjimky. První z nich je výjimka CertificateExpiredException. V případě vyhození této výjimky se v certifikačním řetězu nachází certifikát s prošlou dobou platnosti. Druhou výjimkou je CertificateNotYetValidException. V tomto případě se v certifikačním řetězu nachází certifikát, jehož doba platnosti ještě nenastala. Třída X509TokenFactory

Jedná se o tovární třídu. Tato třída poskytuje funkcionalitu k vytváření instancí certifikátů. Nalézá se v balíčku cz.utb.signlib.util. Tato třída se skládá ze dvou souborů. Prvním souborem je rozhraní CertTokenFactory. Toto rozhraní poskytuje deklaraci metod této třídy.

Druhým souborem je X509TokenFactory. Ten obsahuje třídu, která implementuje dané rozhraní.

### 5.3.10 Metoda createCertToken

V této metodě jsou vytvářeny instance třídy X509Token. Metoda přejímá ve svých parametrech alias, certifikační řetězec, privátní klíč a zdroj certifikátu. Na základě těchto dat vytvoří výše zmíněnou instanci.

## 5.4 Třída X509TokenInterpreter

Tato třída obaluje instanci X509Token, která byla popsána výše. Poskytuje funkcionalitu pro interpretaci certifikátů obsažených v instanci tokenové třídy. Její metody umožňují převést hodnoty obsažené v certifikátech do textového tvaru, který je srozumitelný pro uživatele. Třída se nachází v balíčku cz.utb.signlib.util. Je tvořena jediným souborem X509TokenInterpreter.java.

### 5.4.1 Metoda parsePrincipal

Tato metoda přejímá objekt s informacemi o poskytovateli nebo zřizovateli certifikátu. Na základě tohoto objektu a svých parametrů vrátí jako svou návratovou hodnotu řetězec. V tomto řetězci jsou jednotlivé informace v pořadí, v jakém byly zadány jako parametry metody. Tyto informace jsou odděleny řetězcem znaků, který byl metodě předán jako druhý parametr.

### 5.4.2 Metoda getCertToken

Tato metoda vrací instanci, která je touto třídou obalována.

### 5.4.3 Metoda interpretCertSubjectName

Tato metoda obdrží z certifikačního nejnižší certifikát a metodou parsePrincipal převede informace o zřizovateli certifikátu do textové podoby. Tyto informace vrátí jako svůj výstup.

### 5.4.4 Metoda interpretCASubjectNames

Tato metoda obdrží z certifikátového tokenu certifikáty certifikačních autorit a stejným postupem jako předchozí metoda je převede na řetězce, které vrátí formou pole.

### 5.4.5 Metoda toString

Tato metoda vrací instanci třídy jako textový řetězec. Metoda jako svou návratovou hodnotu vrací výsledek volání výše zmíněné metody interpretCertSubjectName.

## 5.5 Třída `SystemKeyStoreServiceImpl`

Tato třída poskytuje základní funkcionalitu pro přístup k systémovému úložišti certifikátu a umožňuje vyzvednout certifikáty z tohoto úložiště a pracovat s nimi. Tato třída se nachází v balíčku `cz.utb.signlib.system`.

Třída implementuje rozhraní `SystemKeyStoreService`. Toto rozhraní poskytuje deklaraci metod pro práci se systémovým úložištěm certifikátů. Rozhraní rozšiřuje rozhraní `KeyStoreService`, čímž přejímá deklaraci jeho metod.

Třída samotná poskytuje definici metod, deklarovaných v tomto rozhraní. V tomto případě se jedná pouze o metodu `getCertTokens`. Tato metoda zkontroluje, jestli systém poskytuje systémové úložiště certifikátů. V případě že nikoliv, vrátí hodnotu `null`. V opačném případě si vyžádá certifikáty ze systémového úložiště, ty uloží do kolekce typu množina a navrátí je ve své návratové hodnotě.

## 5.6 Třída `FileKeyStoreServiceImpl`

Tato třída se nachází v balíčku `cz.utb.signlib.file`. Poskytuje základní funkcionalitu pro práci se soubory obsahujícími certifikáty. Tato třída implementuje rozhraní `FileKeyStoreService`. Toto rozhraní poskytuje deklaraci metod pro práci s certifikátovými soubory. Toto rozhraní rozšiřuje rozhraní `KeyStoreService`.

Samotná třída poskytuje definici metod, které dané rozhraní deklaruje. V tomto případě se jedná o metodu `loadCertFile`. Třída si uchovává všechny certifikáty, které načte touto metodou ve vlastní kolekci, typu množina. Tuto kolekci vrací metodou `getCertTokens`.

### 5.6.1 Metoda `loadCertFile`

Tato metoda načte certifikátový soubor, jehož cesta byla metodě předána jako první parametr. Následně z tohoto souboru načte všechny certifikáty, které jsou dostupné, a uloží je do úložné kolekce v instanci třídy. Jako druhý argument je předáno metodě heslo, kterým je soubor chráněn.

## 5.7 Třída `SmartCardKeyStoreServiceImpl`

Tato třída poskytuje funkcionalitu pro práci s čipovými kartami a certifikáty na nich uloženými. Je umístěna v balíčku `cz.utb.signlib.smartcard`. Třída implementuje rozhraní `SmartCardKeyStoreService`. To poskytuje deklaraci metod pro práci s čipovými kartami, třídami pro automatickou detekci čteček a čipových karet a pro správu certifikátů umístěných na čipových kartách.

Implementace tohoto rozhraní je třída `SmartCardKeyStoreServiceImpl`. Poskytuje definici k daným metodám. Třída samotná si při své iniciaci vytvoří instance tovární metody pro vytváření certifikátů, tovární metody pro vytváření instancí tříd obalujících knihovny, poskytujících přístup k čipovým kartám a kolekce, ve kterých si ukládá instance vytvořené těmito továrními třídami.

### 5.7.1 Metoda `setUserPkcs11Wrapper`

Tato metoda předá své parametry tovární metodě pro vytváření instancí obalovacích tříd knihoven pro komunikaci s čipovými kartami. Následně vezme návratovou hodnotu a uloží ji do kolekce těchto obalovacích tříd.

### 5.7.2 Metoda `LoadCertCard`

Zde je nejprve zavolána metoda `installValidPkcs11Providers`. Následně je vytvořena instance třídy `KeyStore` nad aplikačním rozhraním `pkcs11`. Poté jsou z instance načteny všechny třídy, které jsou přístupné pod zadaným pinem, který je předán metodě jako první argument. Tyto záznamy jsou poté uloženy do interní skladovací kolekce.

### 5.7.3 Metoda `loadTerminal`

V případě, že je k počítači připojená čtečka čipových karet, nastaví tato metoda interní proměnou instance této třídy indikující přítomnost čtečky čipových karet na hodnotu `true`. A do hodnoty proměnné, která odkazuje na čtečku, nastaví odkaz na tuto čtečku. V opačném případě nastaví hodnotu indikující proměnné na `false`.

#### 5.7.4 Metoda loadCard

Na začátku svého běhu nastaví tato metoda proměnnou indikující přítomnost čipové karty na hodnotu false. Pokud k počítači není připojena čtečka čipových karet, je tato metoda ukončena. V opačném případě je hodnota indikační proměnné přenastavena v závislosti na tom, zda-li je ve čtečce čipová karta či nikoliv.

#### 5.7.5 Metoda isTerminalConnected

Tato metoda vrací hodnotu interní proměnné, která indikuje, zda-li je k počítači připojena čtečka čipových karet.

#### 5.7.6 Metoda isCardPresent

Tato metoda vrací hodnotu interní proměnné, která indikuje, zda-li je ve čtečce čipových karet přítomná čipová karta či nikoliv.

#### 5.7.7 Metoda initSmartCardWatchdogs

Tato metoda inicializuje a spustí instance tříd obstarávajících automatickou detekci čtečky čipových karet a toho, zda-li je do čtečky vložena čipová karta, nebo je z ní vyjmuta. Následně uloží odkazy na tyto instance do patřičných proměnných. Metoda přebírá jako svůj parametr instanci třídy, která poskytuje definici metody pro obsluhu těchto tříd. V případě, že je argument metody roven hodnotě null je metoda ukončena bez jejího provedení.

#### 5.7.8 Metoda stopSmartCardWatchdogs

Tato metoda ověří, jestli se odkazy na instance tříd obstarávajících automatickou detekci čtečky a karty, odkazují na iniciované třídy. V případě, že ano, vyžádá si jejich ukončení.

#### 5.7.9 Metoda eraseKeyStore

Tato metoda vymaže z kolekce udržující certifikáty z čipových karet všechny záznamy. Hlavním důvodem je, že záznam certifikátu se odkazuje na čipovou kartu. Při odebrání čipové karty se tedy všechny záznamy certifikátů stávají neplatnými.

### 5.7.10 Metoda `getPkcs11WrapperList`

Tato metoda poskytuje ve své návratové hodnotě kolekci instancí obalovacích tříd knihoven poskytujících přístup k čipovým kartám. Tato kolekce je typu list. Umožňuje také tyto instance nastavovat.

### 5.7.11 Metoda `pkcs11Wrapper2StringMap`

Tato metoda přebírá jako svůj argument kolekci instancí obalovacích tříd knihoven poskytujících přístup k čipovým kartám. Tuto kolekci předá stejnojmenné metodě tovární třídy a vrátí výsledek volání této metody.

## 5.8 Rozhraní `Pkcs11Wrapper`

Toto rozhraní poskytuje deklaraci metod pro práci s knihovnami, které umožňují přístup k čipovým kartám. Implementací tohoto rozhraní jsou třídy, které umí vytvořit dynamický konfigurační soubor pro danou knihovnu a ten předat třídě „Security“, nebo naopak soubor této třídě odebrat. Tím je zařízeno sestavení poskytovatele služeb pouze z těch knihoven, které jsou pro tento účel vybrány.

### 5.8.1 Metoda `getLibratyPath`

Tato metoda vrací hodnotu interní proměnné, ve které je uložena cesta ke knihovně umožňující komunikaci s čipovou kartou.

### 5.8.2 Metoda `installProvider`

Tato metoda zjistí, zda-li v proměnné odkazující se na knihovnu nachází cesta k souboru. V případě že ano je zavolána metoda `uninstallProvider`. Následně se ověří, zda-li je vnitřní hodnota proměnné indikující povolení nasazení této knihovny nastavena na `true`. V případě, že tomu tak není, je tato metoda ukončena. V případě, že proměnná je nastavena, se sestaví dynamický soubor konfigurace pro danou knihovnu. Následně se tento soubor předá knihovně `Security`.

### 5.8.3 Metoda `isValid`

Tato metoda vrací ve své návratové hodnotě informaci o tom, zda-li interní proměnná instance obsahuje řetězec, který odkazuje na soubor. Pokud je tomu tak, je návratová hodnota této metody `true`. V opačném případě `false`.

### 5.8.4 Metoda `setEnabled`

Tato metoda umožňuje nastavit hodnotu vnitřní proměnné, indikující povolení knihovny, kterou spravuje tato instance. Pokud je argumentem `true`, poté je knihovna kterou tato metoda spravuje povolena. V opačném případě je zakázána.

### 5.8.5 Metoda `isEnabled`

Návratovou hodnotou této metody je stav vnitřní proměnné, která indikuje, zda-li je knihovna, kterou tato třída spravuje, povolena či nikoliv.

### 5.8.6 Metoda `uninstallProvider`

Tato metoda nejprve zjistí, zda-li třída `Security` obsahuje poskytovatele služeb se stejným aliasem, jaký byl vytvořen touto metodou. V případě, že je poskytovatel služeb v knihovně přítomen, je z ní odstraněn.

## 5.9 Třída `CzechEidWrapper`

Tato třída poskytuje definici pro metody z rozhraní `Pkcs11Wrapper`, které implementuje. Jedná se o třídu, která zabezpečuje práci s knihovnou pro přístup k českým občanským průkazům. Třída samotná obsahuje cestu ke knihovně pro práci s těmito kartami a umožňuje její používání.

## 5.10 Třída `OpenSCWrapper`

Tato třída definuje metody pro práci s knihovnou `OpenSC`. Tato knihovna podporuje velké množství čipových karet. Více o tomto projektu je možné se dozvědět na jeho stránkách.

[24]

## 5.11 Třída `UserPkcs11Wrapper`

Tato třída definuje metody z rozhraní `Pkcs11Wrapper`. Její instance obsluhují knihovny, které jsou zadané při běhu aplikace. To umožňuje využívat více čipových karet. Jelikož je možné nadefinovat obalovací třídu pro jakoukoliv knihovnu, která přistupuje ke kartě.

## 5.12 Třída `Pkcs11WrapperFactoryImpl`

Tato třída je tovární metodou, která vytváří instance tříd implementujících rozhraní `Pkcs11Wrapper`. Třída implementuje své rozhraní `Pkcs11WrapperFactory`. To poskytuje základní deklaraci metod pro tuto třídu.

Ve svém těle třída poskytuje definici metod pro generování instancí tříd obalujících knihovny pro přístup k čipovým kartám. Nachází se v balíčku `cz.utb.signlib.smartcard`.

### 5.12.1 Metoda `getOpenSCWrapper`

Návratovou hodnotou této metody je instance třídy, jejíž náplní činnosti je spravovat knihovnu `OpenSC` pro práci s čipovými kartami.

### 5.12.2 Metoda `getCzechEidWrapper`

Tato metoda ve své návratové hodnotě poskytuje instanci třídy `CzechEidWrapper`. Díky ní je možné pracovat s českými občanskými průkazy.

### 5.12.3 Metoda `getUserPkcs11Wrapper`

Tato třída vytváří instanci obalovací třídy pro uživatelsky zadanou knihovnu pro práci s čipovými kartami. Cestu ke knihovně převezme z prvního argumentu. Ve druhém argumentu přebírá alias, který bude knihovnu zastupovat při její identifikaci. Pokud není tento alias dodán třídě ve správné formě, je k sestavení instance použit univerzální alias poskytnutý touto tovární třídou.

### 5.12.4 Metoda `pkcs11Wrapper2StringMap`

Tato metoda přebírá v argumentu kolekci obalovacích tříd pro knihovny přistupující k čipovým kartám. Tato kolekce je převedena na mapu řetězců, z každého řetězce této mapy je možné rekonstruovat původní obalovací třídu.

## 5.13 Třída `CardWatchdogEventImpl`

Tato třída slouží k definování události, která nastane, pokud je do čtečky vložena karta, nebo je-li karta ze čtečky vyjmuta. Tato třída se nachází v balíčce `cz.utb.signlib.smart-card.watchdog`. Toto rozhraní deklaruje pouze metodu, která vrací hodnotu této události.

Samotná třída poskytuje definici této metody, navíc poskytuje tovární metody pro vytvoření jednotlivých událostí, které mohou nastat a metodu nastavovací.

### 5.13.1 Metoda `getEventCardIn`

Jedná se o tovární metodu, která ve své návratové hodnotě poskytne instanci třídy reprezentující událost vložení čipové karty do čtečky.

### 5.13.2 Metoda `getEventCardOut`

Tato metoda ve své návratové hodnotě poskytuje instanci třídy, která reprezentuje událost vyjmutí čipové karty ze čtečky.

### 5.13.3 Metoda `getEvent`

V návratové hodnotě této metody se nalézá hodnota výčtového typu, který reprezentuje událost s používáním čipové karty.

### 5.13.4 Metoda `setEvent`

Tato metoda přiřadí hodnotu výčtového typu reprezentujícího událost, která může nastat ve spojení s čipovou kartou, k instanci této třídy.

## 5.14 Rozhraní `CardWatchdogHandler`

Toto rozhraní slouží k určení třídy, která se stará o události, které nastaly v důsledku vložení nebo vyjmutí čipové karty. Rozhraní poskytuje deklaraci metod pro základní práci s těmito událostmi.

### 5.14.1 Metoda `handleCardWatchdogEvent`

Jedná se o deklaraci metody, která bude zavolána hlídací třídou v okamžiku, kdy nastane událost spojená s čipovou kartou. Třída implementující toto rozhraní musí poskytnout definici této metody, kde patřičnou událost zpracuje.

## 5.15 Rozhraní CardWatchdog

Toto rozhraní deklaruje základní metody pro třídy, které mají dohlížet na události spojené s používáním čipové karty.

### 5.15.1 Metoda addCardWatchdogHandler

Tato metoda přiřadí instanci třídy, která zpracovává událost, která je spojena s čipovou kartou této třídě.

### 5.15.2 Metoda setTerminal

Tato metoda přiřadí do vnitřní proměnné instanci třídy reprezentující čtečku čipových karet. Ta poskytuje důležité metody pro běh těchto dohlížecích tříd.

### 5.15.3 Metoda start

Tato metoda spustí funkci dozorčí třídy, která od té doby sleduje stav čtečky čipových karet. Pokud nastane patřičná událost, spustí obsluhu události na všech přidělených třídách.

### 5.15.4 Metoda stop

Tato metoda ukončuje činnost této instance.

### 5.15.5 Metoda fireWatchDogEvent

Tato metoda zahájí obsluhu události. Metoda projde ve svém těle interní kolekci, která uchovává třídy obsluhující tuto událost. Na všech třídách zavolá patřičnou metodu, která má obsloužit tuto událost.

### 5.15.6 Metoda isRunning

Tato metoda vrátí hodnotu interní proměnné, která signalizuje, zda-li je daná instance této třídy aktivní či nikoliv.

## 5.16 Třída CardInWatchdog

Tato třída slouží jako třída, která obstarává detekci vložení čipové karty do čtečky. V momentě, kdy je karta vložena do čtečky, je spuštěna obslužná funkce na všech dodaných instancích s parametrem „CARD\_IN“.

### 5.17 Třída `CardOutWatchdog`

Tato třída poskytuje definici pro metody obstarávající detekci vyjmutí čipové karty ze čtečky. V momentě, kdy je karta vyjmuta ze čtečky, je spuštěna obslužná funkce na všech dodaných instancích s parametrem „CARD\_OUT“.

### 5.18 Třída `SimpleTerminalWatchdog`

Tato třída poskytuje funkcionalitu pro detekci přítomnosti čtečky čipových karet. Hlídací třída vždy po časovém intervalu volá obslužnou třídu, která detekuje, zda-li se změnil stav připojení terminálu oproti stavu předchozímu.

### 5.19 Balíček `cz.utb.signlib.derivations`

Jelikož je náš svět neustále v pohybu a stále se mění, tak je vysoce pravděpodobné, že i tato knihovna bude potřebovat s postupujícím časem upravit. Ať už se bude jednat o změnu funkcionality, kterou knihovna poskytuje, jelikož časem přibudou nové možnosti a algoritmy programování. Některé algoritmy budou prolomeny a nebude je možné v budoucnu použít. Nebo může jít například o rozšíření funkcionality této knihovny např. pro načítání certifikátu z QR Kódů[25].

Z důvodu zachování celistvosti myšlenky knihovny, je zde vytvořen tento balíček, který poskytuje místo pro uložení nově vzniklých tříd, které budou realizovány, aby nedošlo k narušování původní struktury knihovny. Tento balíček obsahuje jednu ukázkovou třídu poskytující základní podobu odvozené třídy. Tato třída nemá žádnou funkcionalitu.

## 6 VÝSTUPNÍ APLIKACE

Na základě knihovny, kterou jsem vytvořil, jež je popsána v předchozí kapitole, jsem vytvořil aplikaci, která umožňuje nahrávat soubory na server. Na serveru je vypočítána hašovací funkce hash souboru. Ta je předána aplikaci a tato aplikace ji následně podepíše uživatelským klíčem a přidá k tomuto podpisu odpovídající certifikát a vše odešle zpět na server.

V následujících podkapitolách popíši jednotlivé funkce aplikace.

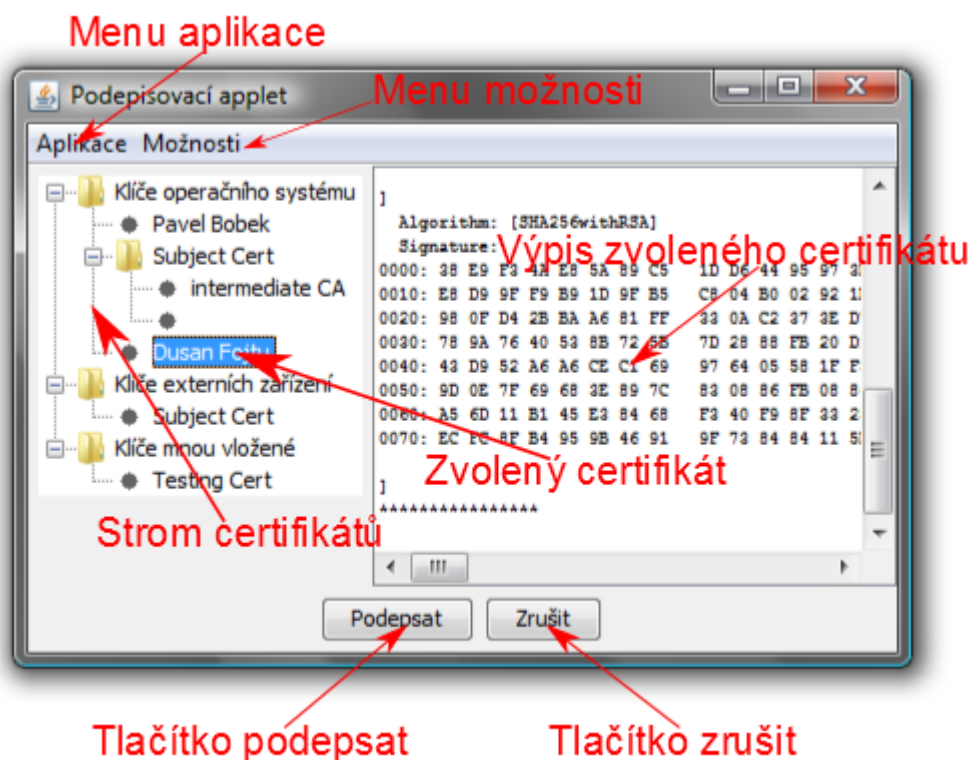
## 6.1 Hlavní okno aplikace

Toto okno je hlavním prostředkem pro komunikaci aplikace s uživatelem. V okně je zobrazen v levé části aplikace strom certifikátů. Tento strom obsahuje veškeré certifikáty dostupné v aplikaci, rozdělené podle jejich původu.

V pravé části je okno, které zobrazuje podrobnosti o právě zvoleném certifikátu. Dále jsou na spodní straně aplikace přítomná dvě tlačítka. První umožňuje podepsat zvolený soubor daným certifikátem. Druhé tlačítko aplikaci ukončuje.

V aplikaci jsou také přítomná dvě menu. První je menu aplikace, které obsahuje pouze možnost ukončit aplikaci z menu. Toto menu je spustitelné klávesovou zkratkou Alt+A. Druhé je menu možnosti. To má přiřazenu klávesovou zkratku Alt+O.

Detail okna aplikace je zobrazen na obrázku (Obr. 6).



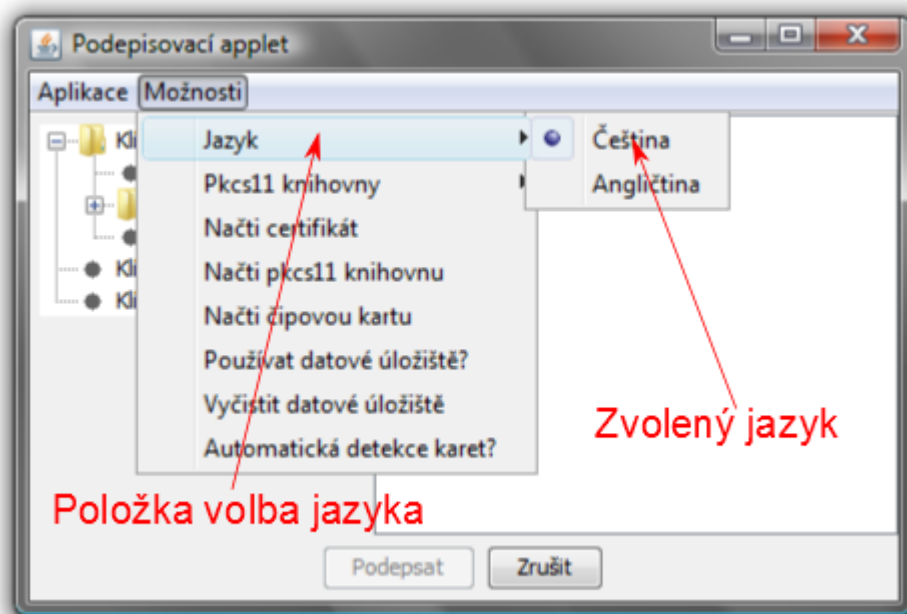
Obr. 6: Hlavní okno aplikace

## 6.2 Menu Možnosti

Toto menu poskytuje hlavní navigaci v aplikaci a umožňuje nastavovat aplikaci, ať už jde o volbu jazyka, nebo o přidání certifikátu do aplikace.

### 6.2.1 Volba jazyka

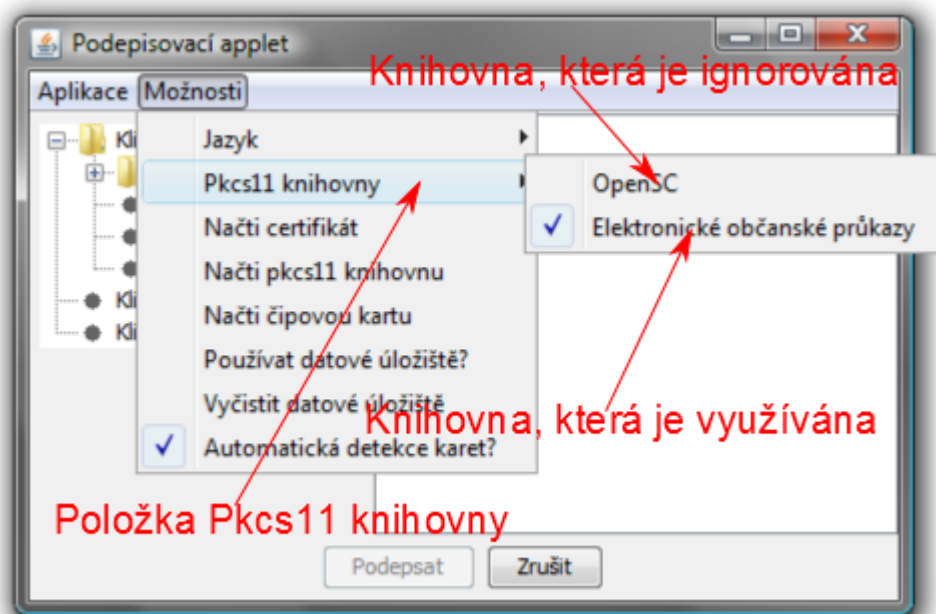
Aplikace umožňuje zvolit jazyk, do jakého je lokalizováno hlavní okno aplikace. V menu možnosti je zde políčko Jazyk (Language), které umožňuje zvolit, v jakém jazyce bude okno aplikace vykresleno. K dispozici jsou jazyky Čeština a Angličtina. Při změně jazyka je okno okamžitě překresleno do nově zvoleného jazyka. Tato část je blíže zdokumentována na obrázku (Obr. 7).



Obr. 7: Volba jazyka v aplikaci

### 6.2.2 Volba Pkcs11 knihovny

Tato volba obsahuje menu s aktuálním výpisem všech knihoven pro přístup k čipovým kartám, které má aplikace k dispozici. Toto podmenu také zobrazuje stav dané knihovny. Pokud je položka knihovny označená, aplikace tuto knihovnu využívá při svém běhu. Pokud položka označená není, aplikace ji při svém běhu ignoruje. Více je popsáno na obrázku (Obr. 8).



Obr. 8: Volba pkcs11 knihovny v aplikaci

### 6.2.3 Volba Načti certifikát

Tato volba uživateli umožňuje načíst soubor obsahující certifikát. Po zvolení této možnosti je uživatel vyzván k zadání cesty k souboru certifikátu a k zadání hesla, kterým je tento soubor chráněn. V případě úspěšného načtení jsou všechny načtené certifikáty zobrazeny v aplikaci.

### 6.2.4 Volba Načti pkcs11 knihovnu

Tato volba umožňuje uživateli zadat cestu ke knihovně implementující rozhraní pkcs#11. Při zvolení této položky je uživateli zobrazeno dialogové okno, ve kterém uživatel zvolí cestu k souboru, který obsahuje knihovnu. Také může zvolit její alias, pod kterým se třída zobrazuje v aplikaci. Pokud uživatel do druhého pole nic nezadá, bude alias vygenerován tovární metodou, která poskytuje obalovací třídu pro tuto knihovnu.

### 6.2.5 Volba Načti čipovou kartu

Tato položka menu umožňuje načíst čipovou kartu do aplikace. Po zvolení této možnosti se uživateli zobrazí dialog. V něm uživatel zadá pin, kterým je karta chráněna. Pokud je pin správný, je připojená čipová karta načtena a všechny její certifikáty uložené pod tímto pinem jsou zobrazeny v aplikaci.

### 6.2.6 Volba Používat datové úložiště

Tato položka nabízí využívání úložiště dat aplikace. Tato aplikace si na počítači uživatele vytvoří soubor, do kterého ukládá patřičná data mezi jednotlivými spuštěními aplikace.

### 6.2.7 Volba Vyčistit datové úložiště

Tato volba umožňuje vymazat datové úložiště, pokud je tato volba nastavena při ukončení aplikace bude datový soubor vymazán a jeho staré hodnoty budou přepsány.

### 6.2.8 Volba Automatická detekce karet?

V případě, že je tato volba zvolena, aplikace využívá automatické detekce vložení či vyjmutí karty. Pokud je karta vložena, aplikace na to zareaguje tím, že zobrazí dialog pro načtení karty. Po vyjmutí karty jsou všechny její certifikáty z aplikace automaticky odstraněny, protože se stali neplatnými.

## 7 TESTOVÁNÍ APLIKACE

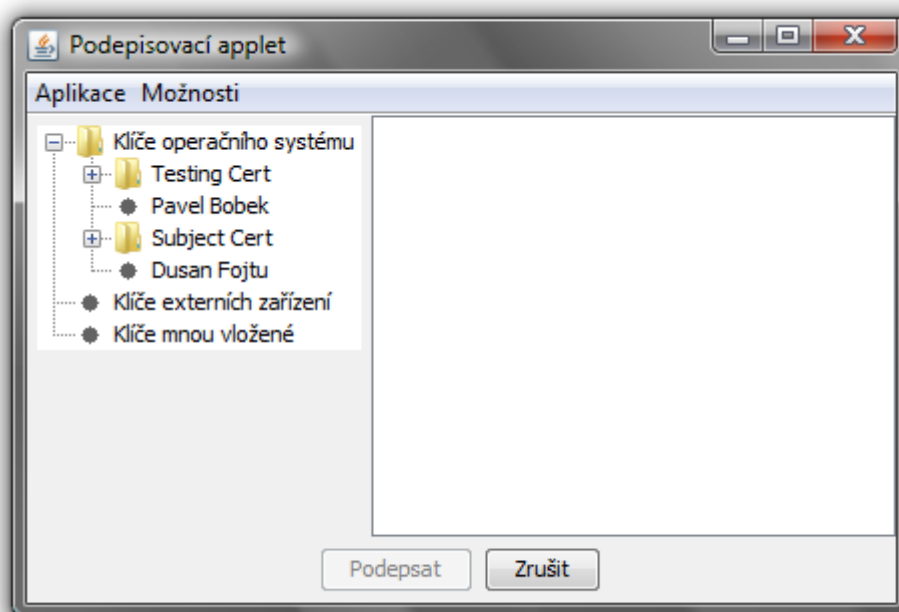
Abych ověřil, zda-li tato aplikace opravdu demonstruje výše uvedené technologie, otestoval jsem ji na daných operačních systémech. Aplikaci jsem nejprve zkompiloval do výstupního jar souboru.

Napsal jsem jednoduchou webovou aplikaci, která umožňuje nahrát soubor do dočasné složky. Následně spočítá hash daného souboru a předá jej appletu. Tento applet podepíše na počítači uživatele. Následně je tento podpis vrácen na server, kde je uložen do databáze.

Tato ukázková aplikace je vystavena na webové adrese „<http://signapplet.php5.cz>“. Ke svému běhu vyžaduje aplikace nainstalovanou Javu 7 a vyšší. Pro použití čipových karet je potřebná čtečka čipových karet a s ní nainstalovaný patřičný ovladač, čipovou kartu a nainstalovanou knihovnu poskytující aplikační rozhraní pkcs#11.

## 7.1 Testování na operačním systému Windows

Jelikož byla aplikace na tomto systému vyvíjena, proběhlo zde její testování bez sebe-menších potíží. Okno se vždy zobrazilo, bylo možné měnit jazyk a přidávat knihovny. Certifikační úložiště bylo načteno se všemi certifikáty. Bylo možné načítat certifikáty ze souboru, knihovny pkcs11, čipové karty, používat datové úložiště a vymazávat jej a používat automatickou detekci karet. Hlavní okno aplikace spuštěné v tomto systému je zobrazeno na obrázku(Obr. 9)



Obr. 9: Zobrazení hlavního okna v operačním systému Windows

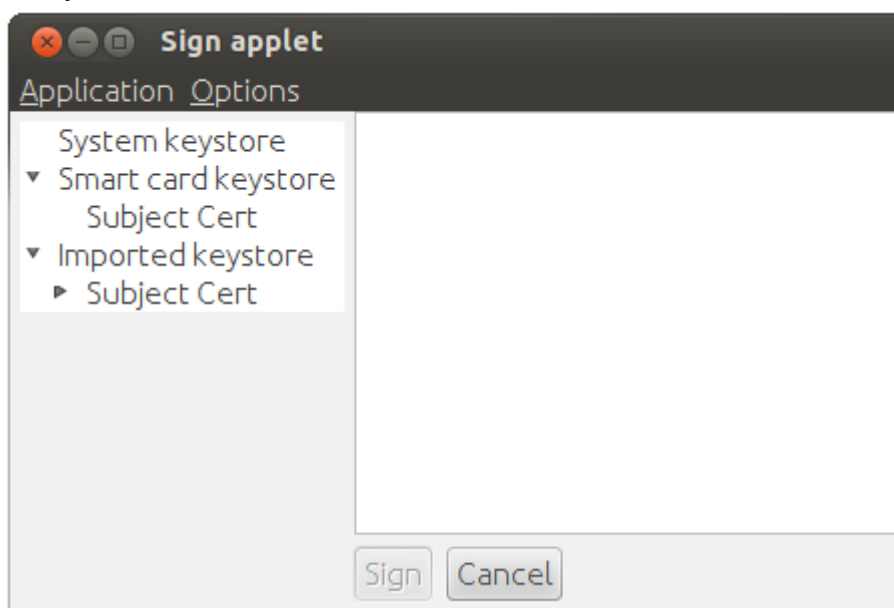
## 7.2 Testování na operačním systému typu Linux

Test proběhl na počítači s operačním systémem ubuntu 12.04 LTS. Aplikace běžela plynule a její komponenty se zdály být funkční. Projevila se chyba v třídách pro automatickou detekci čipové karty.

Po hlubším zkoumání této chyby jsem se dočetl, že chyba je na straně Javy, která neumí korektně pracovat s verzovanými knihovnami. [26]

Jedná se o zásadní chybu, která je na opravitelná pouze z části a nebo je potřeba značných znalostí a administrátorských práv v systému. Proto jsem vyhodnotil, že bude lepší v případě operačního systému Linux odepřít přístup k automatické detekci čipových karet, neboť

realizace kompletního funkčního řešení těchto tříd by byla značně namáhavá a v tomto případě by poskytovala pouze dočasné řešení, jehož selhání by nastalo v době vydání nové verze verzovaných knihoven.



Obr. 10: Zobrazení hlavního okna v operačním systému Linux

Zbytek aplikace pracoval korektně, bez problému se mi podařilo načíst čipovou kartu manuálním způsobem a s její pomocí podepsat soubor. Stejně to bylo i v případě certifikátu načítaného ze souboru. Detail zobrazení hlavního okna je možné vidět v obrázku (Obr. 10).

### 7.3 Testování na operačním systému OS X

Program jsem testoval na školním zařízení. Na tomto počítači byl nainstalovaný operační systém OS X Yosemite v10.10. Při přístupu do systémového úložiště (keychain) bylo nutné nejprve nastavit dostupnost soukromých klíčů, aby byly dostupné pro aplikace. Aplikace načetla tyto klíče z úložiště a podepsala vybraný soubor bez problémů.

K načítání certifikátu ze souborů bylo potřeba, aby aplikace měla přístup do souborového systému. Po nastavení přístupu k souborovému systému v prohlížeči, ve kterém aplikace běžela, bylo možné načíst certifikát a podepsat jím daný soubor.

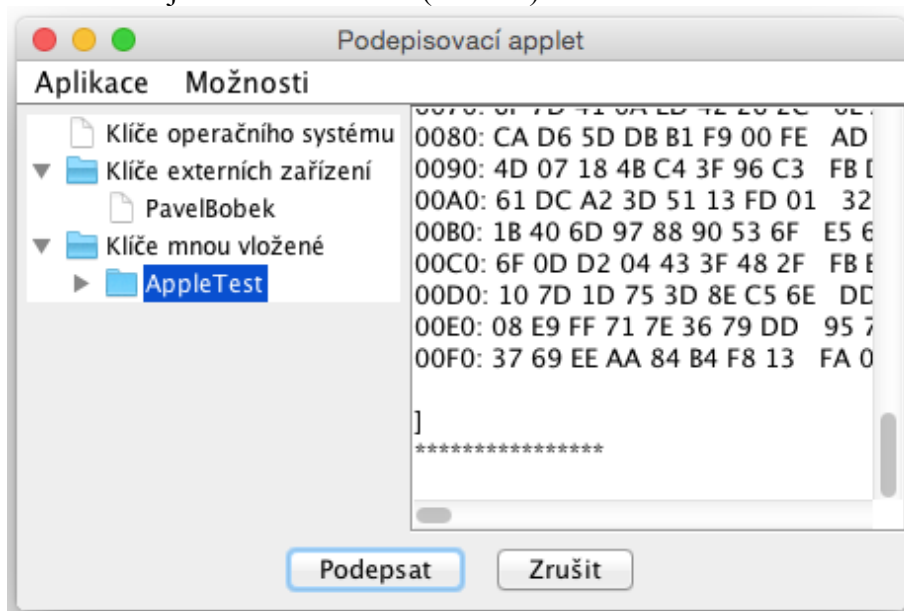
Pro práci s čipovou kartou bylo nutné stáhnout knihovnu poskytující služby čipových karet, jelikož od verze OS X Lion v10.7 není tato podpora součástí systému[27]. Knihovnu bylo možné stáhnout ze stránky „<http://smartcardservices.macosforge.org/trac/wiki/installers>“. Po nainstalování potřebného ovladače ke čtečce čipových karet a knihovny pro přístup k čipové kartě bylo ještě nutné umožnit přístup k soukromému klíči, který je uložený na čipové kartě, skrz aplikaci klíčenka. Následně bylo možné načíst certifikát a soukromý klíč

a podepsat s nimi soubor.

Automatické načítání karet je v tomto systému také nefunkční proto je ponecháno vypnuté.

Všechny ostatní funkce pracovali v systému standardně a bez chyb.

Vzhled hlavního okna je vidět na obrázku (Obr. 11).



Obr. 11: Zobrazení hlavního okna v operačním systému OS X

## ZÁVĚR

V této práci jsem shrnul své poznatky ze studia kryptografie. Věnoval jsem se zde hlavně aplikaci asymetrické kryptografie. Formuloval jsem zde formát certifikátů, které jsou v dnešní době používány jako elektronický podpis.

Dále jsem prostudoval knihovny v balíčku java.security. Zejména jsem se zaměřil na práci těchto knihoven s uvedenými certifikáty. Uvedl jsem zde úložiště, která jsou využívána k jejich skladování a způsob přístupu k těmto úložištím.

Prostudoval jsem způsob podepisování aplikací a uvedl jsem zde, jakým způsobem je možné danou aplikaci podepsat a zvýšit tak její důvěryhodnost.

Na základě zjištěných poznatků jsem naprogramoval knihovnu, která poskytuje přístup k těmto úložištím a umožňuje další práci s těmito certifikáty na nejrozšířenějších platformách (Windows, Linux, Mac OS X).

Funkcionalitu této knihovny jsem demonstroval ukázkovou aplikací. Touto aplikací je applet, který umožňuje uživateli podepisování datových souborů na nejrozšířenějších platformách.

**SEZNAM POUŽITÉ LITERATURY**

- [1] BUDIŠ, Petr. *Elektronický podpis a jeho aplikace v praxi*. Olomouc: ANAG, 2008, s. 7. ISBN 978-80-7263-465-1.
- [2] PIPER, F a Sean MURPHY. *Kryptografie*. Praha: Dokořán, 2006, s. 8-10. ISBN 80-7363-074-5.
- [3] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *The Internet Engineering Task Force*[online]. 2002 [cit. 2015-05-17]. Dostupné z: <https://www.ietf.org/rfc/rfc3280.txt>
- [4] SCHILDT, Herbert. *JAVA 2: Příručka programátora*. Brno: SoftPress, 2001, s. 18-19. ISBN 80-86497-04-6.
- [5] FLANAGAN, David. *Programování v jazyku JAVA*. Praha: Computer Press, 1997, s. 5-10. ISBN 80-85896-78-8.
- [6] Java 8 Information. <https://www.java.com> [online]. 2014 [cit. 2015-05-17]. Dostupné z: <https://www.java.com/en/download/faq/java8.xml>
- [7] Security. <http://docs.oracle.com/> [online]. 2015 [cit. 2015-05-17]. Dostupné z: <http://docs.oracle.com/javase/8/docs/technotes/guides/security/>
- [8] Zobrazení a správa certifikátů. <http://windows.microsoft.com/> [online]. 2015 [cit. 2015-05-17]. Dostupné z: <http://windows.microsoft.com/cs-cz/windows-vista/view-or-manage-your-certificates>
- [9] The SunMSCAPI Provider. <http://docs.oracle.com/> [online]. 2015 [cit. 2015-05-17]. Dostupné z: <http://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html#SunMSCAPI>
- [10] Keychain Services Programming Guide. <https://developer.apple.com> [online]. 2014 [cit. 2015-05-17]. Dostupné z: <https://developer.apple.com/library/ios/documentation/Security/Conceptual/keychainServConcepts/01introduction/introduction.html>
- [11] The Apple Provider. <http://docs.oracle.com/> [online]. 2015 [cit. 2015-05-17]. Dostupné z: <http://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html#Apple>
- [12] <https://wiki.gnome.org/Projects/GnomeKeyring>. <https://wiki.gnome.org/> [online]. 2015 [cit. 2015-05-18]. Dostupné z: <https://wiki.gnome.org/Projects/GnomeKeyring>

- [13] Java Cryptography Architecture Oracle Providers Documentation for JDK 8. [Http://docs.oracle.com/](http://docs.oracle.com/) [online]. 2015 [cit. 2015-05-18]. Dostupné z: <http://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html>
- [14] PKCS #12: Personal Information Exchange Syntax v1.1. [Https://tools.ietf.org/](https://tools.ietf.org/) [online]. 2014 [cit. 2015-05-17]. Dostupné z: <https://tools.ietf.org/html/rfc7292>
- [15] The SunJSSE Provider. [Http://docs.oracle.com/](http://docs.oracle.com/) [online]. 2015 [cit. 2015-05-17]. Dostupné z: <http://docs.oracle.com/javase/8/docs/technotes/guides/security/SunProviders.html#SunJSSEProvider>
- [16] PETERKA, Jiří. *Báječný svět elektronického podpisu*. Praha: CZ.NIC, 2011, s. 178-179. CZ.NIC. ISBN 978-80-904248-3-8.
- [17] ISO 7816. [Http://www.cardwerk.com/](http://www.cardwerk.com/) [online]. 2015 [cit. 2015-05-17]. Dostupné z: [http://www.cardwerk.com/smartcards/smartcard\\_standard\\_ISO7816.aspx](http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx)
- [18] JDK 8 PKCS#11 Reference Guide. [Http://docs.oracle.com/](http://docs.oracle.com/) [online]. 2015 [cit. 2015-05-17]. Dostupné z: <http://docs.oracle.com/javase/8/docs/technotes/guides/security/p11guide.html>
- [19] New security requirements for RIAs in 7u51. COSTLOW, Erik. [Https://blogs.oracle.com](https://blogs.oracle.com) [online]. 2013 [cit. 2015-05-17]. Dostupné z: [https://blogs.oracle.com/java-platform-group/entry/new\\_security\\_requirements\\_for\\_riasp11guide.html](https://blogs.oracle.com/java-platform-group/entry/new_security_requirements_for_riasp11guide.html)
- [20] Working with Signed RIAs. [Http://docs.oracle.com/](http://docs.oracle.com/) [online]. 2015 [cit. 2015-05-18]. Dostupné z: <http://docs.oracle.com/javase/8/docs/technotes/guides/deploy/certificates.html#JSDPG886>
- [21] CODE SIGNING CERTIFICATES. [Http://codesigning.ksoftware.net/](http://codesigning.ksoftware.net/) [online]. 2014 [cit. 2015-05-19]. Dostupné z: <http://codesigning.ksoftware.net/>
- [22] COSTLOW, Erik. Self-signed certificates for a known community. [Https://blogs.oracle.com](https://blogs.oracle.com) [online]. 2013 [cit. 2015-05-17]. Dostupné z: [https://blogs.oracle.com/java-platform-group/entry/self\\_signed\\_certificates\\_for\\_a\\_known\\_community.html](https://blogs.oracle.com/java-platform-group/entry/self_signed_certificates_for_a_known_community.html)
- [23] BLOCH, Joshua. *Java efektivně: 57 zásad softwarového experta*. Praha: Grada, 2001, s. 45-49. ISBN 80-247-0416-1.
- [24] OpenSC [online]. 2015 [cit. 2015-05-17]. Dostupné z: <https://github.com/OpenSC/OpenSC/wiki>
- [25] QRcode [online]. [cit. 2015-05-17]. Dostupné z: <http://www.qrcode.com/en/index.html>

- [26] Oracle, javax.smartcardio failures. ROUSSEAU, Ludovic. *Http://ludovicrouseau.blogspot.cz/* [online]. 2013 [cit. 2015-05-19]. Dostupné z: <http://ludovicrouseau.blogspot.cz/2013/03/oracle-javaxsmartcardio-failures.html>
- [27] SmartCard Services. *Http://smartcardservices.macosforge.org/* [online]. [cit. 2015-05-20]. Dostupné z: <http://smartcardservices.macosforge.org/trac/wiki/installers>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

Obr.	Obrázek
JVM	Virtuální stroj Javy
PKCS	Standardy asymetrické kryptografie
CA	Certifikační autorita

**SEZNAM OBRÁZKŮ**

Obr. 1: Struktura certifikátu X.509v3.....	15
Obr. 2: Občanský průkaz osazený kontaktním čipem.....	23
Obr. 3: Komunikace aplikace s čipovou kartou.....	24
Obr. 4: Příkazy pro aplikaci keytool.....	25
Obr. 5: Struktura knihovny SignLib.....	27
Obr. 6: Hlavní okno aplikace.....	44
Obr. 7: Volba jazyka v aplikaci.....	45
Obr. 8: Volba pkcs11 knihovny v aplikaci.....	46
Obr. 9: Zobrazení hlavního okna v operačním systému Windows.....	48
Obr. 10: Zobrazení hlavního okna v operačním systému Linux.....	49
Obr. 11: Zobrazení hlavního okna v operačním systému OS X.....	50

## SEZNAM TABULEK

## SEZNAM PŘÍLOH

- signlib.zip – archivovaná složka obsahující zdrojové kódy knihovny SignLib
- signlib.jar – zkompileovaná verze výše zmíněné knihovny
- signapp.zip – archivovaná složka obsahující zdrojové kódy aplikace pro podepisování datových souborů.
- signapp.zip – zkompileovaná verze aplikace