Masarykova univerzita
Fakulta informatiky

Integrace RemSig do klientských aplikací

Bakalářská práce

Ondřej Přikryl

Brno, jaro 2016

Místo tohoto listu vložte kopie oficiálního podepsaného zadání práce a prohlášení autora školního díla.

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Ondřej Přikryl

Vedoucí práce: RNDr. Michal Procházka, Ph.D.

Poděkování

Rád bych poděkoval panu RNDr. Michalovi Procházkovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování bakalářské práce.

Shrnutí

Práce se zaměřuje na popis

Klíčová slova

Digitální podpis, RemSig, OAuth 2.0, PKCS#11, CryptoAPI, CNG, CSP, Smart Card Minidriver

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Prvky informačních technologií se stávají každodenní součástí života a většina lidí by si bez nich již nedokázala život představit. Informační technologie uživatelům v mnoha směrech usnadňují život. Slouží jak k práci, tak i ke komunikaci se známými, zábavě a mnoha dalším činnostem. S rozšířením se však zvyšuje i potenciální riziko, které zneužití těchto prvků přináší. V době, kdy internet poskytuje základní komunikaci mezi více než miliardou lidí a je čím dál tím více používán jako nástroj k obchodování, se bezpečnost stává nesmírně důležitou otázkou, která by neměla zůstat nepovšimnuta. Částečnou odpovědí na tuto otázku bylo zavedení kryptografických systémů. I ty však nenabízí řešení ve všech případech a lidé by si měli stále dávat velký pozor.

Kryptografie nebyla původní součástí návrhu internetu, ale v posledních dekádách se díky rozmachu digitálních technologií stala důležitým prvkem internetové infrastruktury. Přestože je skryta před zraky laických uživatelů, jde o důležitý vědní obor, který v současnosti prožívá obrovský posun kupředu. Tento vývoj je způsoben zejména tím, že si uživatelé začínají uvědomovat potřebu chránit své data a soukromí i ve virtuálním světě. Moderní kryptografie má mnoho podob a odvětví, jednou z nich je digitální podpis (též nazýván elektronický podpis), který je nezbytnou součástí systému RemSig.

Systém RemSig je bezpečné vzdálené úložiště certifikátů a privátních klíčů, které uživatelům poskytuje možnost podepisovat dokumenty bez nutnosti neustálého fyzického přístupu k zařízení, na němž je certifikát uložen. Důležitou součástí systému RemSig je uživatelská přívětivost. V současné době je systém možné používat pouze z webového rozhraní INET [1] a IS MU [2] , a proto je nutná implementace knihoven, které integrují funkce RemSigu do klientských aplikací. Náplní mé bakalářské práce je tyto knihovny implementovat a popsat základní prvky CSP a PKCS#11.

Struktura bakalářské práce je následující. První kapitola je věnována úvodu. Druhá kapitola se věnuje základnímu principu digitálních podpisů. Následující kapitola obsahuje popis systému RemSig,

1. Inet MU, viz. `https://inet.muni.cz`.
2. Informační systém Masarykovy univerzity, viz. `https://is.muni.cz`.

jeho vlastnosti a funkcionalitu. Čtvrtá kapitola se zabývá protokolem OAuth 2.0, jeho využitím a schopnostmi. Páta kapitola popisuje standard PKCS#11. Šestá kapitola je poslední kapitolou teoretické části bakalářské práce a věnuje se CryptoAPI, pod kterou spadá většina kryptografie u operačních systémů Microsoft Windows. Sedmá kapitola se zabývá praktickou částí práce. Popisuje implementaci knihoven, úpravy systému RemSig a potřebnou implementaci virtuálního tokenu do budoucna. V osmé kapitole následuje shrnutí výsledků a závěr bakalářské práce.

Očekávaným výsledkem mé bakalářské práce jsou knihovny propojující RemSig a klientské aplikace . Důsledkem toho již není zapotřebí používat webové rozhraní INET a IS MU, což zaměstnancům MU výrazně usnadní práci při podepisování digitálních dokumentů. Nezávislost systému na webovém rozhraní univerzity také umožňuje jeho rozšíření mimo univerzitní kruhy.

2 Digitální podpis

Digitální podpis slouží k nahrazení obyčejného podpisu v digitálním světě. Je založen na asymetrické kryptografii a k jeho použití je zapotřebí dvojice klíčů - soukromého a veřejného. Digitální podpis zaručuje následující vlastnosti:

- Autenticitu - každý podpis je jednoznačně spojen s uživatelskou entitou, kterou může být jak běžný uživatel, tak i organizace nebo státní útvar (digitální značka).
- Integritu dat - zaručuje, že data nebyla po cestě pozměněna. To je velice důležité k ujištění, že data, která byla odeslána, druhá strana také přijala.
- Nepopiratelnost - autor nemůže popřít, že data nepodepsal. To je zajištěno tím, že k vytvoření podpisu je zapotřebí soukromý klíč, který má vlastník podpisu v držení a není veřejně znám.

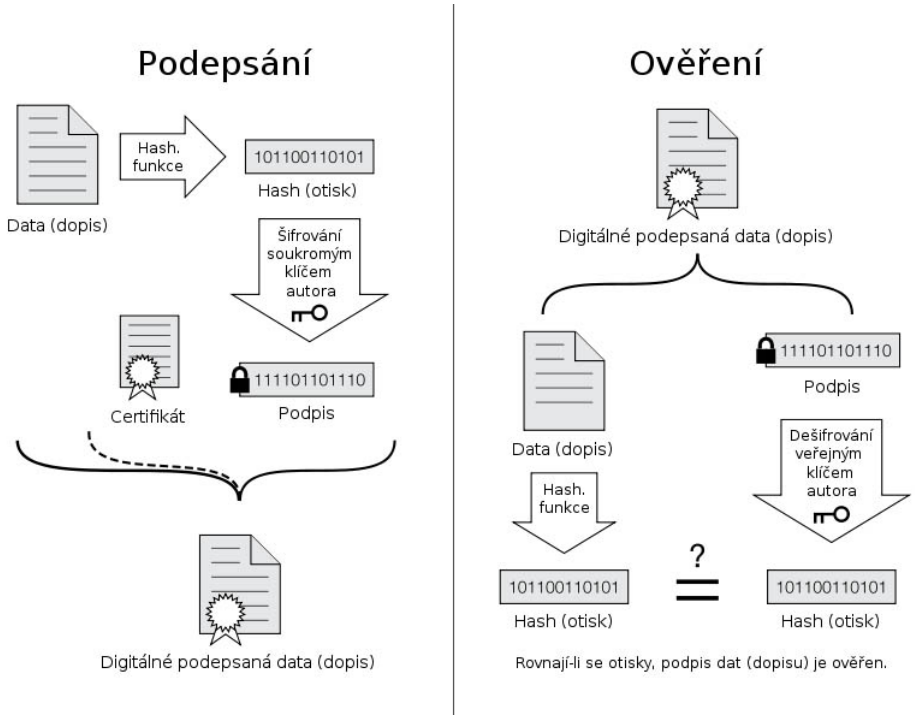
2.1 Princip použití

Uživatel si vytvoří dvojici klíčů, veřejný a soukromý. Soukromý klíč si ponechá a veřejný nechá potvrdit certifikační autoritou a následně jej zvěřejní. Podepsaný veřejný klíč nazýváme ceritifát. Podpis dokumentu poté probíhá následovně:

- uživatel spočítá haš (ang. hash) dokumentu - například pomocí rychlé hašovací funkce SHA256
- haš zašifruje svým soukromým klíčem - například pomocí algoritmu RSA
- data poté odešle společně s certifikátem a zašifrovaným hašem

Ověřování podpisu probíhá analogicky:

- osoba provádějící ověření spočítá haš dokumentu
- za pomoci veřejného klíče osoby která dokument podepsala dešifruje přijatý digitální podpis a ověří, že získaný haš je shodný s vypočteným hašem



Obrázek 2.1: Principy digitálního podpisu.

Pokud si haše odpovídají, pak je jisté, že přijatý dokument je shodný s odeslaným a je tedy zaručena integrita. Aby byla zajištěna nepopiratelnost a autenticita je nutné ověřit, že daný klíč opravdu patří osobě, která poslala podepsaný dokument. Procesu, který toto zajišťuje se říká *přenos důvěry*.

2.2 Přenos důvěry

Aby bylo možné pomocí digitálního podpisu dosáhnout autenticity a nepopiratelnosti, je veřejný klíč dané osoby podepsán důvěryhodnou certifikační autoritou. Tímto způsobem je vyjádřeno, že certifikační autorita věří, že majitel veřejného klíče je skutečně ten za koho se vydává. Uživatel musí certifikační autoritě prokázat svoji totožnost předtím, než získá certifikát. Další možností k dosažení auntenticity je tzv. *síť důvěry*. Jedná se o decentralizovanou alternativu k centralizovanému modelu certifikačních autorit.

3 RemSig

Systém RemSig je bezpečné vzdálené úložiště certifikátů a privátních klíčů, které uživatelům poskytuje možnost podepisovat dokumenty bez nutnosti neustálého fyzického přístupu k zařízení, na němž je certifikát uložen. Díky tomu odpadá uživatelům systému RemSig povinnost nosit s sebou hardwarové zařízení, což zvyšuje uživatelskou přívětivost i funkcionalitu. Uživatelé totiž mohou podepisovat dokumenty, i když u sebe právě nemají token, nebo když jsou na jiném počítači, který není nakonfigurovaný pro práci s digitálními podpisy nebo není důvěryhodný.

3.1 Aktuální využití

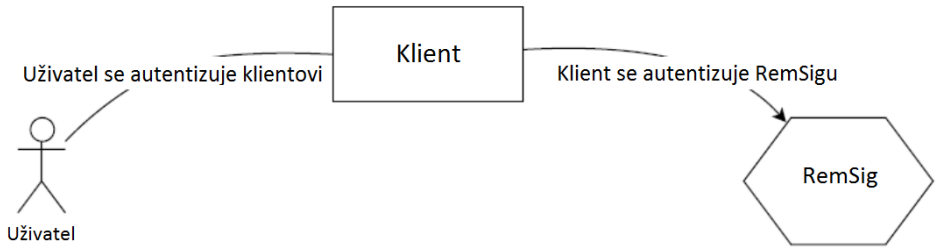
RemSig umožňuje podepisování dokumentů přímo z webového rozhraní systému INET a IS MU a je v souladu se současnou českou legislativou. RemSig je napojen na služby certifikační autority PostSignum [1], což je další výhodou pro zaměstnance MU, kteří tak nemusí řešit vydání certifikátu třetí stranou, ale vše si pohodlně nastaví v rozhraní INET. Aktuálně je používána verze implementovaná v jazyce PHP, ale k dispozici je již nová verze implementovaná v jazyce Java, která již brzy nahradí starou implementaci.

3.2 Autorizace

Pro přístup k API systému je zapotřebí, aby se uživatel autentizoval klientovi, který zprostředkovává komunikaci mezi uživatelem a veřejně přístupnou API RemSigu. Po autentizaci uživatele, Remsig obdrží od klienta unikátní identifikátor, podle kterého určí, ke kterým operacím je klient autorizován. Jestliže se identifikátor nenachází v seznamu pro řízení přístupu (ACL - Access Control List) [2] , klient není oprávněn k žádné operaci.

---

1. Kvalifikovaná certifikační autorita České Republiky, viz. `www.postsignum.cz/`
2. viz. `https://cs.wikipedia.org/wiki/Access_control_list`



Obrázek 3.1: RemSig - Autentizace pomocí klienta.

3.2.1 Autentizace certifikátem

V nové verzi je podporována autentizace certifikátem. Poté, co se uživatel autentizuje klientovi, klient odešle RemSigu certifikát, pomocí kterého RemSig určí, ke kterým operacím je klient oprávněn. Tento přístup však není vyhovovující v případech, kdy chce klient komunikovat přímo s RemSigem. K tomu slouží protokol OpenID Connect.

3.2.2 OpenID Connect

OpenID Connect je již třetí verzí protokolu OpenID. Jedná se o protokol sloužící k ověřování identit, který staví na protokolu OAuth 2.0. OpenID Connect umožňuje klientovi ověřit identitu koncového uživatele a získat základní informace o profilu. Podrobnému popisu protokolu OAuth 2.0 je věnována vlastní kapitola.

3.3 Role

Po úspěšné autentizaci získá klient přístup k metodám odpovídajícím jeho roli. RemSig rozlišuje dva typy rolí:

- Signer - podepisující
- Manager - manažer

Uživatelé systému IS MU mají roli podeposijícího. Ten má přístup k následujícím operacím:

- *sign* - podepíše data podle defaultních kryptografických mechanismů
- *signPKCS7* - vytvoří PKCS#7 [3] podpis dat
- *signPdf* - slouží k podepisování PDF dokumentů

Zatímco manažer má přístup k následujícím operacím:

- *importCertificate* - importuje certifikát do systému RemSig
- *importPKCS12* - importuje strukturu dle standardu PKCS#12 [4]
- *exportPKCS12* - exportuje soukromý klíč a certifikát ve struktuře definované standardem PKCS#12
- *changePassword* - změní heslo, kterým je chráněn soukromý klíč
- *changeCertificateStatus* - změní status certifikátu
- *listCertificates* - vypíše všechny certifikáty uložené v systému RemSig pro danného uživatele
- *checkPassword* - zjistí, zda-li poskytnuté heslo skutečně slouží pro odemknutí privátního klíče

3.4 Funkcionalita

Volání metod probíhá pomocí veřejně dostupné API. Při požadavku se klient musí autentizovat pomocí přístupového tokenu nebo certifikátu. Parametry a data jsou posílány formou XML dokumentu pomocí metody HTTP POST. Data odeslána RemSigu na podpis musí být kódována metodou BASE64[5], přijatá data jsou také kódována pomocí BASE64. Kvůli bezpečnosti je použit protokol HTTPS.

---

3. viz. `http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-7-cryptographic-message-syntax-standar.htm`
4. viz. `http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs12-personal-information-exchange-syntax-standard.htm`
5. viz. `https://cs.wikipedia.org/wiki/Base64`

```
Input XML:
<?xml version="1.0"?>
<remsig>
        <person>
                <uco>1234</uco>
        </person>
</remsig>
```

Obrázek 3.2: Ukázka XML požadavku metody *listCertificates*.

```
OutputXML:
<?xml version="1.0"?>
<remsig>
        <certificate id="1">
                ...    - certificate informations
        </certificate>
        <certificate id="2">
                ...
        </certificate>
        <operationId>1111</operationId>
</remsig>
```

Obrázek 3.3: Ukázka XML odpovědi metody *listCertificates*.

3.4.1 Správa úložiště

Aby uživatel mohl používat systém RemSig, musí v něm mít uložen certifikát a soukromý klíč. Remsig umožňuje generování dvojice soukromého a veřejného klíče přímo v systému. Po vygenerování je soukromý klíč zašifrován pomocí přiloženého PINu, který si uživatel sám zvolí. Dvojice je poté uložena do úložiště a klientovi je nazpět zaslán veřejný klíč, který je potřeba podepsat certifikační autoritou. Podepsaný certifikát je nutno opět odeslat systému. Uživatel však nemusí použít systém pro generování nové dvojice, ale může importovat již existující dvojici ve struktuře PKCS#12 do systému.

- Import certifikátu

  Metoda *importCertificate* slouží k importování certifikátu do úložiště. Certifikát musí být ve formátu PEM. Tato metoda je používána v případě, kdy byla dvojice klíčů vygenerována systémem RemSig a veřejný klíč byl předán na podepsání certifikační autoritou.

- Import struktury PKCS#12

  Metoda *importPKCS12* slouží k nahrání dvojice vytvořené mimo systém RemSig. Tato dvojice je uložena ve struktuře PKCS#12, která je zabezpečena heslem. Poté, co uživatel zadá heslo, je ze struktury exportována dvojice a uložena do úložiště RemSig. Uživatel musí poskytnout nové heslo, kterým bude zašifrován soukromý klíč.

- Export struktury PKCS#12

  Metoda *exportPKCS12* slouží k exportování dvojice ze systému RemSig. Jedinou možností jak exportovat certifikát a soukromý klíč je vytvořením PKCS#12 struktury, do které je po zadání uživatelova PINu k soukromému klíči uložen certifikát a dešifrovaný soukromý klíč. Celá struktura je posléze zamknuta heslem.

3.4.2 Podepisování dat

Jestliže má uživatel k dispozici certifikát a soukromý klíč, může začít používat metody vzdáleného podpisu. RemSig má k dispozici 3 metody na podepisování dat.

- Defaultní podpis

  Metoda *sign* slouží k podpisu dat. K podpisu je použit defaultní kryptografický mechanismus.

- PKCS#7 podpis

  Digitální podpisy se od sebe mohou lišit v použitých algoritmech, ve formátu výstupu a v tom, zda-li výsledný podpis má být přiložen k datům či má být obdržen samostatně.

Metoda *signPKCS7* umožňuje nastavit tyto hodnoty a vytvořit podpis dle standardu PKCS#7. Systém RemSig používá k nastavení těchto hodnot tzv. *prifily*.

Profily
Jestliže má organizace specifické požadavky na PKCS#7 podpis, může si vytvořit profil, ve kterém si nastaví požadované hodnoty. Při vytváření nového profilu musí být nastaveny následující atributy:

- algorithm - algoritmus, který bude použit k podpisu
- no-detach - určuje, zda-li bude podpis přiložen k datům
- encoding - určuje formát výstupu

```
<ceskaposta_01>
<algorithm>SHA1withRSA</algorithm>
<nodetach>TRUE</nodetach>
<encoding>PEM</encoding>
</ceskaposta_01>
```

Obrázek 3.4: Profil české pošty.

Na obrázku je uveden profil české pošty s označením jedna. Metoda s tímto profilem použije k vytvoření haše dat algoritmus SHA1 a výsledný haš zašifruje pomocí algoritmu RSA, podpis bude přiložen společně s daty a výstup bude ve formátu PEM.

- Podpis PDF dokumentu

  Pro podepisování PDF dokumentů slouží metoda *signPdf*. Uživatel si zde může zvolit, zda bude vložen vodoznak do dokumentu či ne. K dispozici je již přednastavený vodoznak pro dokumenty MU. Uživatel si však může vytvořit vlastní nastavení, ve kterém je možné uvést pozici vodoznaku, stránky na kterých bude vodoznak umístěn, obrázek pozadí vodoznaku a text, který bude obsahovat.

4 OAuth 2.0

OAuth 2.0 (RFC 6749) je autorizační protokol (resp. framework), který umožňuje aplikacím třetích stran získat přístup ke službám HTTP, a to buď jménem registrovaného uživatele anebo jménem aplikace třetí strany. Protokol specifikuje pouze autentizaci klienta a výdej přístupového tokenu, přístup k jednotlivým zdrojům je již zcela nezávislý na protokolu. Je tedy zapotřebí veřejně dostupné API. Protokol OAuth 2.0 nahrazuje starý protokol OAuth 1.0 a není s ním zpětně kompatibilní.

4.1 Motivace

V tradičním modelu autentizace klient-server, žádá-li uživatel o přístup k zabezpečenému obsahu, je zapotřebí jeho autentizace pomocí přihlašovacích údajů. Aby tedy aplikace třetí strany mohla přistoupit k témuž obsahu, je zapotřebí s ní sdílet přihlašovací údaje oprávněné osoby. To však s sebou přináší několik problémů:

- Aplikace třetích stran si uchovávají přihlašovací údaje k budoucímu použití. Údaje bývají většinou uchovávány v otevřené podobě. Uchovávání pouze otisku je nedostatečné.
- Uživatel ve většině případů nemůže omezit přístup aplikace k chráněnému obsahu, a to jak časovou platností, tak i omezením obsahu, ke kterému získá aplikace přístup.
- Obvykle lze odebrat přístup aplikaci k zabezpečenému obsahu pouze změnou hesla. To se však projeví i u dalších aplikací, které jsou závislé na stejných přihlašovacích údajích.
- Kompromitace aplikace třetí strany znamená ohrožení přihlašovacích údajů uživatele a všech souborů na nich závislých.

OAuth tyto problémy řeší zavedením autorizační vrstvy a oddělením rolí klienta (aplikace) a vlastníka chráněného zdroje. V protokolu aplikace zažádá o přístup ke chráněnému obsahu a jsou jí přiděleny přihlašovací údaje (přístupový token) odlišné od přihlašovacích údajů uživatele.

4.2 Role

OAuth 2.0 definuje 4 typy rolí:

- Vlastník zdrojů:
  Entita schopná přidělit přístup ke chráněnému obsahu. Jedná-li se o osobu, nazýváme ji koncový uživatel.
- Server zdrojů:
  Poskytovatel a hostitel chráněného zdroje, schopný obsluhovat požadavky ke chráněnému zdroji pomocí přístupového tokenu.
- Klient:
  Aplikace, která vyžaduje přístup ke chráněnému obsahu.
- Server zdrojů:
  Server, který vydává klientovi přístupový token po úspěšné autentizaci oprávněné osoby.

Protokol nijak nespecifikuje interakci mezi autorizačním serverem a serverem zdrojů. Může se jednat o nezávislé entity, ale i o jeden server.

4.3 Přístupový token

Pokaždé když klient potřebuje přistoupit k chráněnému obsahu, potřebuje přístupový token (anglicky *access token*). Přístupový token je řetězec znaků, který opravňuje klienta k přístupu k chráněnému obsahu. V řetězci jsou skrytě uloženy informace o délce platnosti tokenu, o rozsahu přístupu k chráněnému obsahu a informace o vlastníkovi obsahu. Délka platnosti se může lišit implementací, ale typicky bývá platnost tokenu nastavena na 1 hodinu (3600 sekund).

4.4 Obnovující token

Obnovující token (anglicky *refresh token*) je získán společně s přístupovým tokenem a slouží k obnovení přístupového tokenu po vypršení jeho platnosti. Možnost získání přístupového tokenu pomocí obnovujícího tokenu bývá ve většině případech omezena počtem.

4.5 Průběh protokolu

OAuth definuje několik základních způsobů autorizace vedoucích k získání přístupového tokenu. Patří mezi ně:

- Autorizačním kódem:
  Jedná se o doporučený průběh, ke kterému je zapotřebí uživatelský agent (prohlížeč). Klient s použitím prohlížeče přesměruje uživatele na přihlašovací stránku. Po úspěšné autentizaci zašle autorizační server klientovi autorizační kód. Klient pro získání přístupového tokenu zašle autorizační kód autorizačnímu serveru.

- Implicitně:
  Tento typ je zjednodušením průběhu autorizace autorizačním kódem. Slouží pro aplikace běžící v prohlížeči, které používají skriptovací jazyky jako je například JavaScript. Narozdíl od předchozího typu, klient získá přístupový token přímo po přihlášení. Tento typ je rychlejší, jelikož redukuje počet výzev a odpovědí, které je potřeba udělat pro získání tokenu.

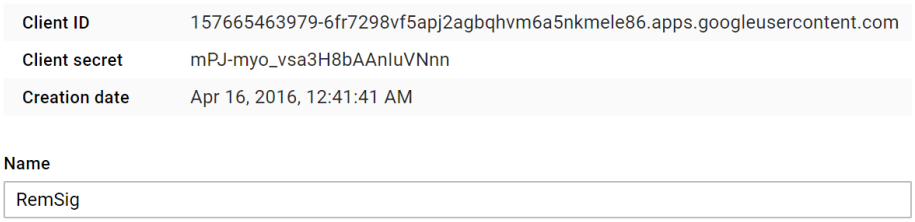
- Zasláním přihlašovacích údajů:
  V tomto typu uživatel poskytne přihlašovací údaje klientovi. Klient zašle přihlašovací údaje autorizačnímu serveru, který po úspěšné autentizaci uživatele zašle zpět přístupový token. Přihlašovací údaje jsou použity jen pro získání přístupového a obnovujícího tokenu, není tedy potřeba jejich uložení pro budoucí použití. Tento typ je používán pouze při vysoké důvěře mezi uživatelem a aplikací.

4.6 Registrace klienta

Předtím než začne klient používat protokol, je potřeba jeho registrace na autorizačním serveru. Pro zvýšení bezpečnosti by klient měl poskytnout:

- jeho typ - webová aplikace, nativní aplikace, zařízení
- adresu, na kterou bude token přesměrován
- jakoukoliv jinou informaci, kterou autorizační server požaduje (název, popis, smluvní podmínky, atd...).

Poté co se klient úspěšně zaregistruje, obdrží od autorizačního serveru unikátní identifikátor a heslo. Tyto údaje jsou nutné pro autentizaci klienta serveru během průběhu protokolu. Je-li při průběhu vyžadována adresa přesměrování, tak musí souhlasit s adresou uvedenou klientem při registraci.



Obrázek 4.1: Google OAuth - registrace klienta.

5 PKCS#11

S nástupem kryptografie přišla nutnost zavést jednotné protokoly a postupy pro zajištění kompatibility koncových aplikací. Ačkoliv se vývojáři shodli na základních kryptografických postupech, kompatibilita mezi jednotlivými implementacemi nebyla nikdy zaručena. Jednotný standard byl tedy nutností. Tuto potřebu naplnila organizace RSA Laboritories zavedením jednotného standardu Public-Key Cryptography Standards, zkráceně PKCS. Jedná se o celou rodinu standardů mezi které patří například i PKCS#11, který je využit v implementaci knihoven RemSig.

PKCS#11 je platformově nezávislé aplikační programové rozhraní (API) pro kryptografické tokeny, nazývané Cryptoki podle Cryptographic Token Interface. PKCS#11 je nyní ve verzi 2.20 a to již od roku 2004. S ohledem na stáří protokolu probíhá od roku 2009 testování nové verze protokolu s označením 2.30, který však ještě nebyl standardizován.

5.1 Cíle návrhu

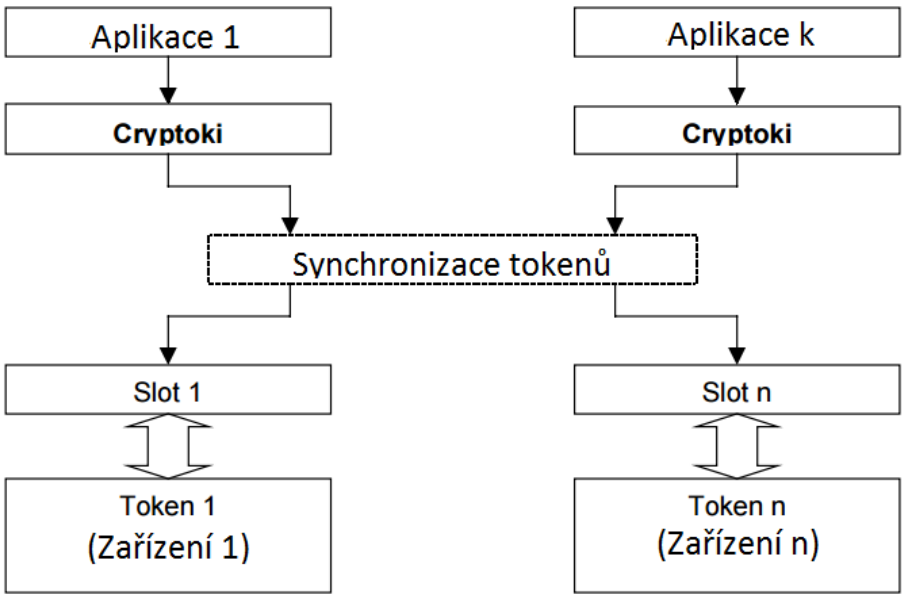
Cryptoki byl navržen s důrazem na jednotnost přístupu k rozdílným zařízením. Cílem rohraní je poskytnout vyšší aplikační vrstvě shodný přístup ke kryptografickému zařízení bez ohledu na konkrétní typ zařízení. Může se tak jednat o Smart Card [1] , PCMCIA [2] kartu nebo síťovou službu, aplikace využívající Cryptoki s nimi může pracovat bez ohledu na jejich hardwarovou odlišnost.

Druhotným cílem bylo sdílení prostředků. S rozvojem víceúlohových operačních systémů se stalo žádoucím být schopen sdílet jedno zařízení mezi více aplikacemi. Stejně tak jedna aplikace by měla být schopna operovat s více kryptografickými zařízeními.

1. viz. `https://cs.wikipedia.org/wiki/Smart_Card`
2. viz. `https://cs.wikipedia.org/wiki/PCMCIA`

5.2 Model návrhu

Model použití Cryptoki začíná jednou nebo více aplikacemi, které požadují přístup k jednomu nebo více kryptografických zařízením. Tyto zařízení provedou operace požadované aplikací a vrátí výsledek.



Obrázek 5.1: PKCS11 - Obecný model.

Cryptoki je tedy rozhraní mezi vyšší vrstvou uživatelských aplikací a nižší hardwarovou vrstvou, která vykonává kryptografické operace. Jak bylo popsáno výše, tato hardwarová vrstva může být reprezentována kartami, tokeny nebo dokonce vzdálenými síťovými službami.

Díky tomu, že Cryptoki maskuje skutečnou hardwarovou podobu připojeného zařízení a zobrazuje je jako totožné jednotky, volající aplikace nemusí znát přesné rozhraní ke kterému je zařízení připojeno či jaký ovladač pro ně použít.

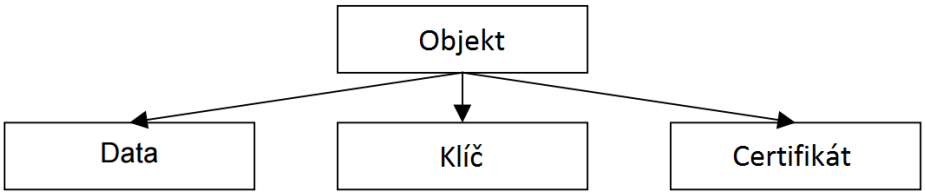
Je nutné podotknout, že Cryptoki je pouze standard, nikoli knihovna. Tímto standardem je definované pouze rozhraní poskytované aplika-

cím, nikoliv podporované vlastnosti. Tyto se mohou lišit podle dané implementace a navíc standard nepředpisuje jaké vlastnosti mají být implementovány.

5.2.1 Objekty

Cryptoki vidí token jako zařízení, na kterém jsou uchovávány objekty a může vykonávat kryptografické operace. Cryptoki definuje 3 základní typy objektů:

- data - jsou definovány a spravovány aplikacemi
- certifikáty - uchovává certifikáty veřejného klíče
- klíče - uchovává kryptografické klíče



Obrázek 5.2: PKCS11 - Objekty.

Objekty jsou zároveň klasifikovány dle jejich životnosti a viditelnosti. Podle životnosti jsou objekty rozděleny na *session* a *token* objekty. *Token objekty* jsou objekty viditelné všem aplikacím připojeným k tokenu a jejich životnost není ovlivněna žádnou relací (tzn. setrvávají na tokenu dokud nejsou odstraněny). Zatímco *session objekty* jsou viditelné pouze aplikaci, která je vytvořila a zanikají s ukončením relace. Objekty se dále mohou dělit na soukromé a veřejné. Pro přístup k veřejnému objektu se aplikace nemusí autentizovat tokenu, zatímco u soukromých je požadováno přihlášení pomocí PINu nebo jiné autentizační metody (např. biometrická zařízení).

5.2.2 Uživatelé

Cryptoki definuje dva typy uživatelů:

- Security Officer (dále již jen SO)
- normální uživatel

Oba vykonávají v systému různé úlohy, ale standard nedefinuje jejich vzájemný vztah. Je tedy v pořádku, když je stejná osoba normálnímm uživatelem i SO.

Účelem SO je inicializovat token a nastavit normálnímu uživateli PIN nebo jinou formu autentizace. SO, na rozdíl od normálního uživatele, dokáže přistoupit pouze k veřejným objektům a nikoli k soukromým.

Normální uživatel je jediný kdo může přistupovat k soukromým objektům, ale až poté co se úspěšně autentizuje. Autentizace normálního uživatele může proběhnout až poté, co SO nastaví uživateli jeho PIN.

5.2.3 Relace

Pro práci s funkcemi a objekty tokenu je potřeba vytvořit alespoň jednu relaci (ang. *session*). Jedná se o logické spojení, ve kterém aplikace komunikuje s tokenem. Relace se dělí na dva základní druhy:

- R/O relace (pouze pro čtení)
- R/W relace (pro čtení i zápis).

R/W relace umožňují aplikaci číst, vytvářet, modifikovat a mazat objekty uložené na tokenu, zatímco R/O relace má přístup pouze ke čtení těchto objektů. Toto omezení se však vztahuje pouze na *token objekty*, nikoliv na objekty vytvořené relací, ke kterým mají oba dva typy plný přístup. Po vytvoření relace má aplikace přístup pouze k veřejným objektům. Aby relace poskytovala přístup k soukromým objektům tokenu, je nutné aby se uživatel autentizoval tokenu.

Cryptoki umožňuje aplikacím vytvářet k jednomu tokenu více relací, ale zároveň také umožňuje tokenu omezit počet R/O a R/W relací, které může aplikace využít. Při zániku relace jsou odstraněny všechny *session objekty* této relace a to i v případě, že jsou tyto objekty

používány jinou relací.

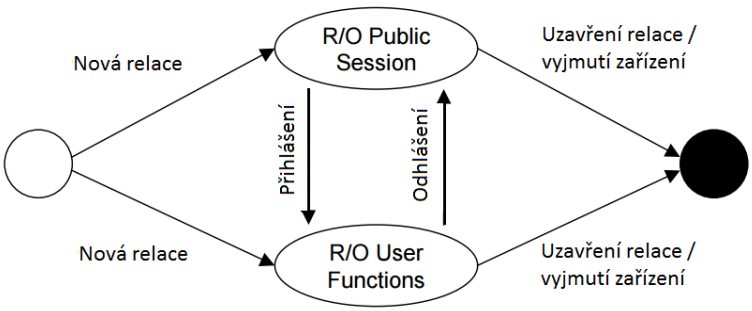
Stavy
Otevřená relace může být v jednom z pěti možných stavů. Tyto stavy definují, které objekty budou dostupné a jaké operace na nich budou povoleny. Pro R/O relace platí:

- R/O Public Session

  Aplikace si otevřela R/O relaci a doposud není autentizována tokenu. Aplikace má právo ke čtení veřejných *token objektů* a právo ke čtení/zápisu veřejných *session objektů*.

- R/O User Functions

  Aplikace si otevřela R/O relaci a uživatel se autentizoval tokenu. Aplikace má přístup ke čtení všech *token objektů* a právo ke čtení/zápisu všech *session objeků*.



Obrázek 5.3: PKCS11 - Objekty.

- R/W Public Session

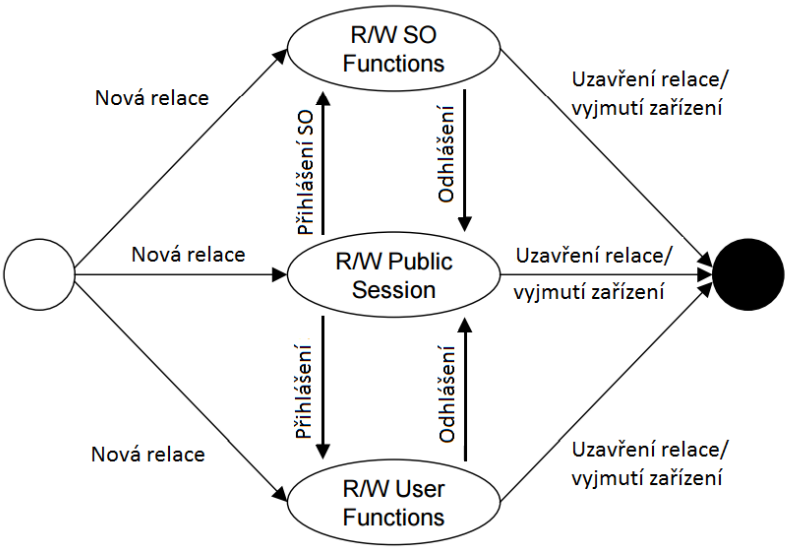
  Aplikace si otevřela R/W relaci a doposud není autentizována tokenu. Aplikace má právo ke čtení/zápisu všech veřejných objektů.

- R/W User Functions

  Aplikace si otevřela R/W relaci a uživatel se autentizoval tokenu. Aplikace má přístup ke čtení/zápisu všech objektů.

- R/W SO Functions

  Aplikace si otevřela R/W relaci a SO se autentizoval tokenu. Aplikace má přístup pouze ke čtení/zápisu všech veřejných *token objeků*. SO může nastavit PIN uživateli.



Obrázek 5.4: PKCS11 - Objekty.

6 CryptoAPI

Microsoft CryptoAPI je aplikační programové rozhraní (též nazývané CAPI) dostupné v operačních systémech Microsoft Windows. Jedná se o skupinu knihoven, které implementují kryptografické mechanismy. CryptoAPI se poprvé objevilo v operačním systému Windows NT 4.0 a v dalších verzích OS bylo postupně zdokonalováno. API podporuje jak symetrickou tak i asymetrickou kryptografii a poskytuje řadu užitečných kryptogafických funkcí jako například kryptograficky bezpečný generátor pseudonáhodných čísel.

S operačním systémem Windows Vista však přišel nový model s názvem Cryptography API: Next Generation (zkráceně CNG). Předpokládá se, že tento model v dlouhodobém časovém horizontu nahradí CryptoAPI. CNG rozšiřuje původní implementaci o nové algoritmy a funkce. V novém modelu například přibyla kryptografie využívající eliptických křivek, která nabízí stejnou úroveň zabezpečení při použití kratšího klíče, což ji činí efektivnější než algoritmus RSA. V modelu přibyli také nová knihovna BaseCSP a Smart Card minidriver, které usnadňují výrobcům čipových karet integraci funkcionality jejich karet do operačního systému Windows.

6.1 CSP

Výrobci mohou použít funkcionalitu kryptografických rozhraní imlementováním speciálního modulu. Tyto moduly se nazývají Cryptographic Service Providers (zkráceně CSP). Jedná se o *.dll* soubory, které jsou určeny k práci s kryptografickými funkcemi. CSP může implementovat veškerou funkcionalitu definovanou kryptografickými rozhraními od symetrické kryptografie po práci s klíči a úložištěm certifikátů systému Windows.

6.2 BaseCSP

S příchodem nového kryptografického modelu Cryptography API: Next Generation přibyla také nová knihovna s názvem BaseCSP. Tato knihovna usnadňuje výrobcům čipových karet integraci funkcionality

jejich karet do operačního systému Windows. Tím pádem již není potřeba budovat celé řešení skrze CSP, ale implementovat pouze menší část pomocí *Smart Card Minidriveru*. Samotné BaseCSP poskytuje minidriveru rozsáhlou funkcionalitu, stará se o práci s certifikáty v úložišti certifikátů operačního systému Windows (anglicky označován jako *CertStore*), umožňuje hašování dat pomocí mnoha dostupných hašovacích algoritmů a další.

6.3 CNG KSP

Narozdíl od CryptoAPI, Cryptography API: Next Generation rozděluje CSP na 2 základní prvky a to:

- Key Storage Provider (zkráceně KSP)
- Cryptographic Algorithm Provider (zkráceně CAP)

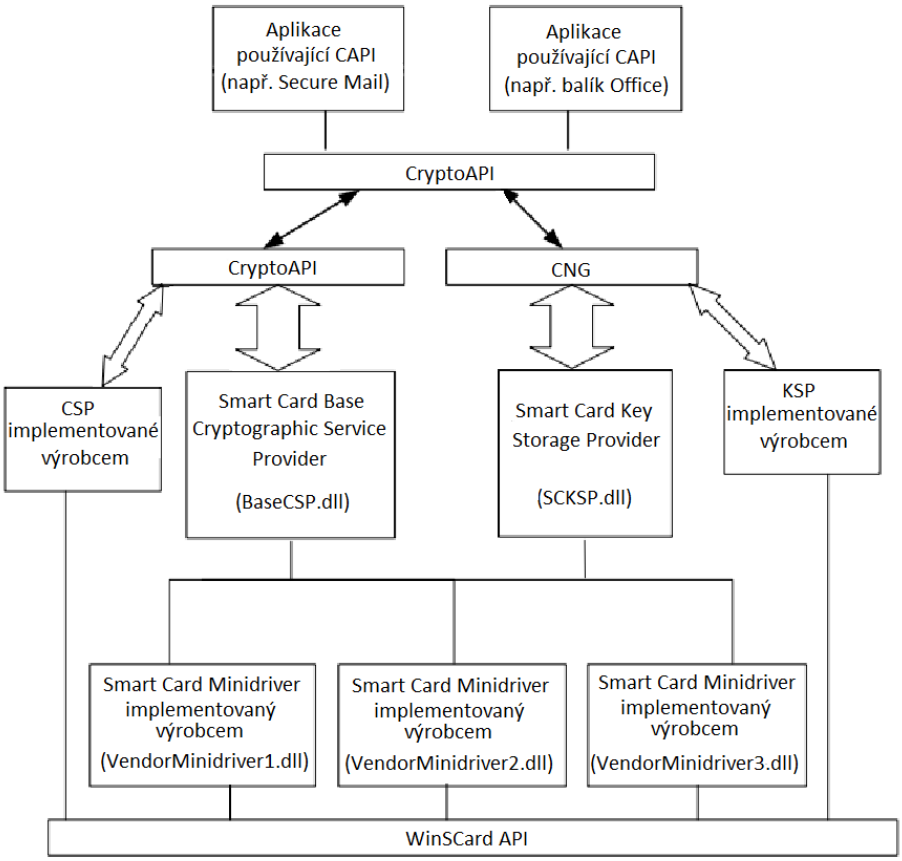
KSP je používán pro veškerou práci s úložištěm klíčů, zároveň však obsahuje i základní algoritmy pro hašování a podepisování dat.

6.4 Smart Card Minidriver

Smart card minidriver je alternativní cestu budování kryptografického modulu narozdíl od implementace celého CSP. Jelikož používá funkce již dostupných BaseCSP nebo KSP, jeho implementace je velmi jednodušší a efektivnější. BaseCSP/KSP se stará o veškeré dění v rámci úložiště certifikátů, nabízí mnoho hašovacích funkcí a další.

Minidriver je dostupný od operačního systému Windows Vista. V průběhu vývoje minidriveru byla přidávána nová funkcionalita, knihovna má nyní verzi 7.07, s posledními aktualizacemi k datu 25. února 2016. Starší nebo neaktualizované operační systémy mohou mít nižší verzy, avšak pro splnění základní funkcionality musí být podporována verze alespoň verze 4.

Na následujícím obrázku je uveden obecný model celého systému. Minidriver je zobrazen jako jedna ze tří alternativ implementace modulu pro práci s kryptografickými zařízeními.



Obrázek 6.1: Role minidriveru v systému.

6.4.1 Role

Specifikace minidriveru definuje 3 základní typy rolí:

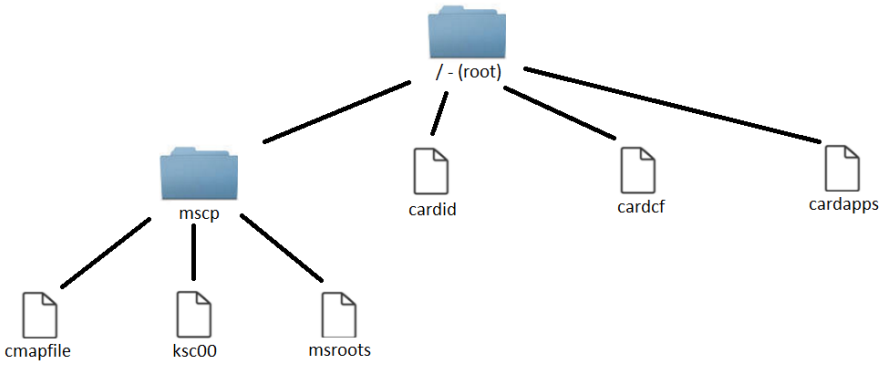
- admin
- uživatel
- kdokoliv

Úlohou administrátora je správa karty. Admin je jediný, kdo může inicializovat a smazat souborový systém. Dále má vlastnost odblokovat

PIN uživatele a jako jediný může nastavit kartě stav "pouze pro čtení". Role uživatele slouží pro práci s certifikáty a klíči, jako je vytváření, upravování a mazání kontejnerů a digitální podpis dat. Nepřihlášená entita má přístup pouze ke čtení veřejných souborů karty.

6.4.2 File system

Každé zařízení, které použije minidriver musí mít specifický souborový systém. V případě, že karta používá jiný systém práce s daty, musí soubory emulovat.



Obrázek 6.2: Souborový systém minidriveru.

- /cardid - soubor obsahuje 16 bytový identifikátor karty.
- /cardcf - uchovává cache karty. Kdykoliv je na kartě něco změněno, hodnota v souboru se zvýší, čímž je indikováno, že si systém musí znovu načíst všechny soubory.
- /cardapps - obsahuje 8 bytový název aplikační podsložky, pro CryptoAPI je název složky "mscp".
- /mscp/cmapfile - tento soubor obsahuje informace o kontejnerech, které jsou na kartě k dispozici.
- /mscp/ksc00 - soubor s digitálním certifikátem ve formátu DER.
- /mscp/msroots - obsahuje strom certifikačních autorit.

6.4.3 Kontejnery

CryptoApi a CNG používají pro ukládání klíčů takzvané *kontejnery*. Model definuje 2 typy asymetrických klíčů:

- Signature Only (klíč určený k digitálnímu podpisu dat)
- Key Exchange (sloužící k výměně klíčů)

Do každého kontejneru může být uložen maximálně 1 klíč od každé typu. Jestliže je do kontejneru přidán klíč stejného typu, původní klíč je přemazán.

6.4.4 Práce s registry

Pro každé kryptografické zařízení, které má být spuštěno pod minidriverem se musí vytvořit sada záznámů v registru. Záznamy jsou přidány do HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Calais\SmartCards\NázevVýrobce

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Calais\SmartCards\RemSig]
"80000001"="minidriver.dll"
"ATR"=hex:3B,B7,94,00,81,31,FE,65,53,50,4B,32,33,90,00,D1
"ATRMask"=hex:ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff,ff
"Crypto Provider"="Microsoft Base Smart Card Crypto Provider"
"Smart Card Key Storage Provider"="Microsoft Smart Card Key Storage Provider"
```

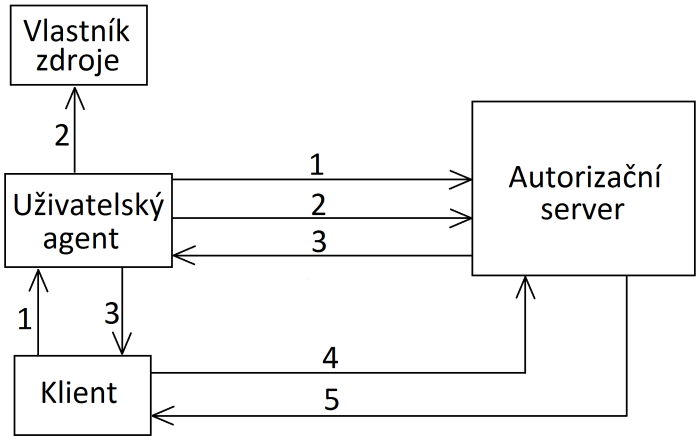
Obrázek 6.3: Registrační soubor minidriveru.

Atribut ATR slouží v tomto kontextu k identifikaci karty, zatímco atribut ATRMask určuje, které části předchozího atributu jsou důležité a podle kterých WinSCard identifikuje zařízení. Atribut 80000001 určuje, že se jedná o minidriver a předává cestu k implementovanému modulu. Následují údaje, které určují, zda-li je podporováno načtení pod BaseCSP, popřípadě KSP.

7 Praktická část

7.1 OAuth 2.0

V praktické části jsem implementoval funkce standardního způsobu autorizace autorizačním kódem. Parametry jsou zasílané metodou POST.



Obrázek 7.1: OAuth 2.0 - Autorizace autorizačním kódem.

1. Klient pomocí prohlížeče přesměruje vlastníka zdroje na autorizační server. Klient v požadavku uvede svůj identifikátor, požadovaný rozsah přístupu, identifikační řetězec [1] a adresu, na kterou má být zaslán autorizační kód.
2. Vlastník zdroje se autentizuje serveru a potvrdí klientovi přístup o daném rozsahu.
3. Autorizační server přesměruje prohlížeč na adresu obdrženou v prvním kroku a předá jí autorizační kód a identifikační řetězec obdržený v prvním kroku.

1. Identifikační řetězec (tzv. state) slouží k identifikaci sezení při větším počtu různých OAuth požadavků.

4. Klient zažádá autorizační server o přístupový token. V požadavku uvede autorizační kód přijatý v předchozím kroku, identifikátor klienta, příslušné heslo a adresu použitou k získání autorizačního kódu v prním kroku.

5. Autorizační server zkontroluje identifikační údaje klienta a ověří platnost autorizačního kódu. Jestliže jsou všechny údaje platné, autorizační server zašle přístupový a obnovující token klientovi.

7.2 Úpravy systému RemSig

RemSig v nynější podobě rozeznává dvě základní role. Role *Signer* je určena pouze pro podepisování dat, zatímco role *Manager* pouze pro správu systému. Implementace knihoven však využívá funkcionalitu obou rolí. Z tohoto důvodu bude vytvořena nová role, která bude obsahovat následující funkce:

- *listMyCertificates* - vypíše certifikáty přihlášeného uživatele
- *checkPassword* - zkontroluje heslo k privátnímu klíči
- *sign* - vytvoří digitální podpis

Nová role bude určena přímo pro knihovny integrující RemSig do klientských aplikací. Jelikož je jako autentizační metoda těchto knihoven použit protokol OAuth a pro server je možné získat z přístupového tokenu edentitu uživatele, všem výše uvedeným metodám je odebrán parameter *"uco"*, který sloužil k identifikaci uživatele.

7.3 PKCS#11 modul

V druhé části praktické práce jsem implementovat PKCS#11 modul, který slouží k propojení systému RemSig a aplikací, které modul PKCS#11 podporují. Mezi tyto aplikace patří například Mozilla Firefox, Safari, Mozilla Thunderbird, Evolution a mnoho dalších [2]. Standard PKCS#11 nepředpisuje, aby všechny jeho funkce byly implementovány,

2. Seznam podporovaných aplikací: `https://github.com/OpenSC/OpenSC/wiki/Using-smart-cards-with-applications`.

avšak definuje, které funkce by měli být implementovány, aby modul měl nějakou funkčnost. S ohledem na systém RemSig byl vytvořen profil PKCS#11, který umožňuje práci s digitálním podpisem. Virtuální zařízení - token, je zobrazováno jako zařízení pouze pro čtení a tím pádem všechny funkce, které vytvářejí, upravují nebo mažou objekty nejsou podporovány. Každý aktivní token je již inicializován a uživatel má nastavené přihlašovací heslo. Aktivní token zároveň obsahuje právě jeden certifikát a jeden privátní klíč, k danému certifikátu.

7.3.1 Obecné funkce

C_Initialize - inicializuje celý modul a vytvoří kontext klientské strany, který slouží k uchovávání informací o modulu. Tato funkce dále nastavuje logování a získává certifikáty z úložiště RemSig. Ke každému certifikátu obdrženému metodou *listMyCertificates* je vytvořen virtuální token, který je posléze připojen do virtuálního slotu.

C_Finalize - tato funkce je volána při ukončení práce s modulem. Slouží k uvolnění kontextu a k uzavření souboru logování.

C_GetFunctionList - obsahuje seznam všech implementovaných funkcí modulu.

C_GetInfo - poskytuje informace o modulu. Je zde uvedena použitá verze cryptoki a název, výrobce a verze modulu.

7.3.2 Správa modulu

C_GetSlotList - slouží k získání seznamu dostupných slotů. Aplikace může určit, zda-li má být navrácen kompletní seznam nebo pouze seznam slotů, ve kterých je přítomen token.

C_GetSlotInfo - představuje základní informace o každém statickém slotu.

C_GetTokenInfo - slouží k získání informací o tokenu. Funkce nastaví tokenu hodnoty popis, seriové číslo a vydavatel podle certifikátu, který obsahuje. Dále každému tokenu nastaví následující vlastnosti:

- *CKF_WRITE_PROTECTED* - token je pouze pro čtení.
- *CFK_LOGIN_REQUIRED* - uživatel musí být přihlášen pro přístup k některým objektům a funkcím.
- *CKF_USER_PIN_INITIALIZED* - PIN uživatele je již nastaven, zabrání volání funkce *C_InitPin*.
- *CKF_TOKEN_INITIALIZED* - token je již inicializován, zabrání volání funkce *C_InitToken*.

C_GetMechanismList - funkce vrací seznam podporovaných kryptografických mechanismů tokenu. V seznamu se nyní nachází pouze jeden prvek a to *CKM_SHA256_RSA_PKCS*, který odpovídá základnímu mechanismu metody *sign* používané systémem RemSig.

C_GetMechanismInfo - slouží k získání podrobných informací o mechanismu.

7.3.3 Správa relací

C_OpenSession - vytvoří novou relaci mezi aplikací a tokenem. Jestliže má token již otevřenou existující relaci s aplikací, synchronizuje jejich stavy.

C_CloseSession - uzavře relaci a uvolní paměť.

C_CloseAllSessions - ukončí všechny relace a uvolní paměť, je volána před ukončením práce s modulem.

C_GetSessionInfo - poskytuje informace o relaci, její stav, ID slotu, ke kterému je vytvořena a další.

C_Login - autentizuje uživatele tokenu. Při úspěšném přihlášení, funkce nastaví uživateli příslušnou roli, čímž uživatel získá přístup k privátním objektům a funkcím. Všechny relace, které souvisí se stejným tokenem jsou následně aktualizovány do přihlášeného stavu. PIN je uložen do kontextu tokenu.

C_Logout - odhlásí uživatele a uvolní paměť. Všechny relace jsou aktualizovány do nepřihlášeného stavu.

7.3.4 Práce s objekty a digitální podpis

C_FindObjectsInit - inicializuje vyhledávání objektů, poskytnutá šablona je uložena do kontextu relace, která použila funkci.

C_FindObjects - zahájí samotné vyhledávání objektů. Každý token vlastní jeden certifikát a k němu jeden privátní klíč. Jestliže je objekt nalezen, funkce vrací identifikátor objektu.

C_FindObjectsFinal - ukončí operaci vyhledávání a uvolní šablonu z relačního kontextu.

C_GetAttributeValue - funkce slouží k získání atributu objektu, opět je poskytnuta šablona, která určuje, které atributy jsou požadovány.

C_SignInit - autentizuje uživatele tokenu. Při úspěšném přihlášení, funkce nastaví uživateli příslušnou roli, čímž uživatel získá přístup k privátním objektům a funkcím. Všechny relace, které souvisí se stejným tokenem jsou následně aktualizovány do přihlášeného stavu.

C_Sign - odhlásí uživatele a uvolní paměť.

7.3.5 Práce s modulem

Jestliže chce aplikace pracovat s modulem, pak musí pro jeho plnou funkčnost splnit sérii příkazů. Typickým příkladem práce aplikace s modulem je následující posloupnost:

1. Uživatel musí získat přístupový token pomocí OAuth aplikace. Jestliže token není k dispozici, modul se nepodaří inicializovat.
2. Aplikace inicializuje modul, který získá certifikáty ze systému RemSig.
3. Poté si aplikace vylistuje dostupné tokeny a vybere si token, se kterým chce pracovat. Aplikace si může získat informace o tokenu a mechanismech, které podporuje.

4. Pro práci s tokenem si aplikace vytvoří jednu nebo více relací pomocí funkce *C_OpenSession*.

5. V tomto stavu je možné vyhledat veřejné objekty tokenu. Pro přístup k privátním objektům je zapotřebí se přihlásit pomocí funkce *C_Login*. PIN k tokenu je odeslán systému RemSig pro ověření správnosti. Po úspěšném přihlášení jsou aktualizovány všechny relace do přihlášeného stavu.

6. Aplikace inicializuje vyhledávání objektů pomocí funkce *C_FindObjectsInit*, přičemž funkci předá šablonu, která specifikuje hledaný objekt.

7. Následuje samotné vyhledávání objektů. Funkce *C_FindObjects* získá identifikátory objektů, které se shodují s parametry šablony.

8. Vyhledávání je ukončeno zavoláním funkce *C_FindObjectsFinal*, paměť obsahující šablonu je uvolněna.

9. Aplikace vykoná kryptografické operace, v našem případě digitální podpis dat. Pro použití jeného tokenu není potřeba opět inicializovat modul, ale lze začít od kroku číslo 3.

10. Při ukončení práce s modulem jsou zavolány funkce *C_CloseSessions* nebo *C_CloseAllSessions*, které ukončí relace mezi aplikací a tokenem a následně *C_Finalize*, který uvolní kontext modulu.

7.4 Smard card minidriver

Ve třetí části praktické práce jsem implementoval Smart card minidriver, který slouží k propojení systému RemSig a úložiště certifikátu operačního systému Windows. Vytvořený minidriver podporuje načtení jak pod Smart Card BaseCSP, tak i pod CNG Key Service Provider. Každý certifikát v systému Windows je zobrazován jako samostatný token, který má vlastnost pouze pro čtení. Dokumentace minidriveru přesně specifikuje, které funkce musí být implementovány jestliže je token pouze pro čtení.

7.4.1 Inicializace a zánik

CardAcquireContext - inicializuje komunikaci mezi BaseCSP/KSP a minidriverem. BaseCSP/KSP předá minidriveru strukturu obsahující informace o kartě (jméno, ATR), o požadované verzy knihovny a o funkcích, které slouží pro práci s daty (alokace paměti, uvolnění paměti, cache). Jestliže je karta minidriverem rozpoznána, funkce vytvoří kontext karty a pomocí metody *listMyCertificates* získá certifikát ze systému RemSig a uloží si jej do kontextu. V poslední řadě funkce přidá do struktury tabulku funkcí, které minidriver implementuje. Minimální podporovaná verze minidriveru je verze 4.

CardQueryCapabilities - získá přehled možností karty. V aktuální verzi jsou definovány pouze 2 vlastnosti - možnost karty generovat klíče, kterou karta pouze pro čtení nedisponuje a zda-li karta provádí vlastní kompresi dat (není nutná komprese v baseCSP/KSP). Nastavením této hodnoty na pravda zamezíme kompresi souborů zvenčí.

CardGetProperty - slouží k získání vlastností karty. Touto operací lze získat vlastnosti jako jsou například seriové číslo karty, podporované kryptografické mechanismy a informace, zda je karta pouze pro čtení.

CardDeleteContext - tato funkce je volána při ukončení práce s minidriverem. Slouží k uvolnění veškerého kontextu, který minidriver alokoval.

7.4.2 Přihlášení a odhlášení

CardAuthenticatePin - autentizuje uživatele tokenu. Heslo k tokenu je zkontrolováno pomocí metody *checkPassword*. Při úspěšném přihlášení, funkce nastaví uživateli příslušnou roli, čímž uživatel získá přístup k privátním objektům a funkcím.

CardDeauthenticate - odhlásí uživatele a uvolní paměť. Tato funkce je definována jako velitelná.

CardAuthenticateEx - tato funkce nahrazuje funkci *CardAuthenticatePin*, která je určena pro nižší verze. Tato funkce kromě klasického

přihlášení heslem přidává podporu externí autentizace (např. pomocí biometriky), autentizace typu výzva/odpověď a další. Implementace minidriveru podporuje pouze klasické přihlášení pomocí hesla. Heslo k tokenu je opět zkontrolováno pomocí metody *checkPassword*. Při úspěšném přihlášení, funkce nastaví uživateli příslušnou roli, čímž uživatel získá přístup k privátním objektům a funkcím.

CardDeauthenticateEx - tato funkce nahrazuje funkci *CardDeauthenticate*, která je určena pro nižší verze. Funkce odhlásí uživatele a uvolní paměť. Jestliže není dostupná a je vyžadováno odhlášení, BaseCSP/KSP se pokusí restartovat kartu.

7.4.3 Operace s veřejnými daty

Jelikož minidriver pracuje s virtuálním tokenem, je zapotřebí virtualizovat souborový systém. V implementaci jsou zahrnuty všechny podstatné soubory pro správný chod minidriveru dle jeho specifikace.

CardEnumFiles - slouží k získání řetězce souborů ve složce určené parametrem.

CardReadFile - získá obsah požadovaného souboru.

CardGetFileInfo - získá informace o velikosti a přístupových právech souboru.

CardQueryFreeSpace - funkce indikuje, kolik je na kartě kontejnerů, určuje maximální počet kontejnerů a velikost volného místa. Jelikož je karta pouze pro čtení, k dispozici není žádné volné místo a aktuální a maximální počet kontejnerů je 1.

7.4.4 Kontejnery

CardGetContainerInfo - slouží k obdržení veřejného klíče kontejneru. Klíč musí být ve formátu DER.

CardGetContainerProperty - získá vlastnosti kontejneru, jedná se o alternativní funkci k *CardGetContainerInfo*.

CardQueryKeySizes - vrací délku veřejných klíčů, které jsou podporované zařízením.

7.4.5 Kryptografické operace

CardSignData - provede digitální podpis dat pomocí metody *sign* systému RemSig.

7.5 Virtualizace zařízení v operačním systému Windows

Proces objevení karty je nedílnou součástí práce s kryptografickými zařízeními v operačním systému Windows, který nenabízí tak vysokou úroveň abstrakce jako PKCS#11 modul. Na samém dnu modelu se nachází knihovna Winscard.dll, která slouží k přiřazení karty k odpovídajícímu minidriveru. Po připojení karty je získán jednoznačný identifikátor karty (nazván ATR), který se operační systém pokusí vyhledat v systémovém registru, v němž jsou uloženy záznamy o kryptografických zařízeních, jejich identifikátorech a příslušných cestách k odpovídajícím minidriverům.

Jelikož se v návrhu žádné fyzické zařízení nevyskytuje, je zapotřebí vytvořit ovladač k virtuální čtečce karet a simulovat vložení karty do systému, což vyvolá signál ke spuštění minidriveru. Při hledání této problematiky jsem našel spoustu komerčních modelů, které využívají výšše popsaný princip simulace vložení karty, ovšem žádný volně šířitelný aktuální model. Díky nemožnosti virtualizace karty nebyl implementovaný minidriver nijak testován a jedná se pouze o prototyp.

7.6 Budoucí rozšíření knihoven a poznámky k systému RemSig

Knihovny nyní podporují jen základní metodu pro podpis dat systémem RemSig. Tato metoda vytvoří haš dat za využití hašovací funkce SHA256 a haš posléze zašifruje soukromým klíčem pomocí algoritmu RSA. Do knihoven by bylo možné přidat podporu PKCS#7 profilů,

které systém RemSig podporuje motodou *signPKCS7*.

Bezpečnostní token je zařízení, které může vlastnit více certifikátů a privátních klíčů, které jsou chráněné jedním společným heslem (heslem zařízení). Současná implementace systému RemSig však přiřazuje každému privátnímu klíči samostatné heslo, čímž úložiště představuje n tokenů pro n dvojic privátních klíčů a certifikátů. Vzhledem k velkému počtu certifikátů v úložišti by mohli být certifikáty rozděleny do několika skupin (např. komerční, atd...), které by byly chráněny jednotným heslem. Tímto by se rapidně snížil počet virtuálních tokenů simulovaných v knihovně PKCS#11 a virtualizovaných pro minidriver.

7.7 Externí závislosti

7.7.1 Libcurl

Libcurl je multiplatformní klientská knihovna sloužící k přenosu dat, jejíž výhody jsou například jednoduchost použití, množství podporovaných protokolů a možnost použití zdarma. Mezi podporované protokoly patří FTP, HTTP, HTTPS, LDAP a mnohé další. Podporované platformy zahrnují mezi jinými také Solaris, NetBSD, Mac OS X a Linux. Klíčovými vlastnostmi knihovny, které ji činí jednoduchou k použití je podpora vícevláken, IPv6 a důkladná dokumentace.

7.7.2 OpenSSL

OpenSSL je open source projekt, který poskytuje knihovny implementující základní kryptografické funkce. Projekt je napsán v jazyce C, ale poskytuje mnoho tzv. "wrapperů", díky kterým je možné knihovny používat i v ostatních programovacích jazycích. Aktuální verze je 1.0.2 vydána k 22. lednu 2015, ale v průběhu tohoto roku je očekávána nová implementace s označením 1.1.0. OpenSSL je platformově nezávislé.

7.7.3 Libxml2

Libxml2 je nástroj vyvinutý pro platformu Gnome, sloužící k práci s XML soubory. Navzdory původu v Gnome je použitelný i v jiných prostředích včetně Windows a MacOS. Co si z Gnome zachovává je pou-

žitelnost zdarma, obsáhlá dokumentace, široká podpora a uživatelská základna a v neposlední řadě kompatibilita s mnoha programovacími jazyky.

8 Závěr

9 Bibliografie

[1] Adams, C., Lloyd, S. *Understanding public-key infrastructure: concepts, standards, and deployment considerations*. Indianapolis: New Riders Publishing, 1999, 296 p. ISBN 157870166X.

[2] Vigašová, S. *Client Tools for RemSig* [online]. Brno, Masarykova univerzita, 2015 [cit. 20. 5. 2016]. 48 s. Dostupné na: `https://is.muni.cz/auth/th/409781/fi_b/vigasova_bp.pdf`

[3] Hardt, D. *The OAuth 2.0 Authorization Framework* [online]. Internet Engineering Task Force, 2012[cit. 20. 5. 2016]. 76 s. Dostupné na: `http://tools.ietf.org/html/rfc6749`

[4] Jirůtka, J. *OAuth 2.0* [online]. Praha, Wiki FIT ČVUT, 26. 2. 2014, [akt. 30. 3. 2016], [cit. 20. 5. 2016]. Dostupné na: `https://rozvoj.fit.cvut.cz/Main/oauth2`

[5] *PKCS #11 v2.20: Cryptographic Token Interface Standard* [online]. RSA Laboratories, 2004 [cit. 20. 5. 2016]. 407 s. Dostupné na: `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf`

[6] Kubina, T. *Vzdialené PKCS#11 úložisko* [online]. Brno, Masarykova univerzita, 2010 [cit. 20. 5. 2016]. 56 s. Dostupné na: `https://is.muni.cz/auth/th/172593/fi_m/dp.pdf`

[7] Scholz, F. *PKCS11 Implement* [online]. Mozilla Developer Network and individual contributors, 13. 7. 2007, [akt. 7. 5. 2014], [cit. 20. 5. 2016]. Dostupné na: `https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/PKCS11_Implement`

[8] Tarasov, V., Rousseau, L. *OpenSC – tools and libraries for smart cards* [online]. GitHub, 11. 12. 2012, [akt. 15. 1. 2016], [cit. 20. 5. 2016]. Dostupné na: `https://github.com/OpenSC/OpenSC/wiki`

[9] Coleridge, R. *The Cryptography API, or How to Keep a Secret* [online]. Microsoft Developer Network Technology Group, 19. 8.

1996, [cit. 20. 5. 2016]. Dostupné na: https://msdn.microsoft.com/en-us/library/ms867086.aspx

[10] *Cryptography API: Next Generation* [online]. Microsoft Developer Network, [cit. 20. 5. 2016]. Dostupné na: https://msdn.microsoft.com/en-us/library/windows/desktop/aa376210(v=vs.85).aspx

[11] *CryptoAPI Cryptographic Service Providers* [online]. Microsoft Developer Network, [cit. 20. 5. 2016]. Dostupné na: https://msdn.microsoft.com/en-us/library/windows/desktop/bb931357(v=vs.85).aspx

[12] Mysore, S. *Smart Card Base Cryptographic Service Provider (Base CSP)* [online]. Microsoft Developer Network Blog, 30. 11. 2005, [cit. 20. 5. 2016]. Dostupné na: https://blogs.msdn.microsoft.com/shivaram/2005/11/30/smart-card-base-cryptographic-service-provider-base-csp/

[13] *Windows Smart Card Minidriver Specification* [online]. Microsoft Developer Network, 9. 7. 2009, [akt. 25. 2. 2016], [cit. 20. 5. 2016]. Dostupné na: http://download.microsoft.com/download/3/3/2/332FD70B-F04D-470A-A135-040350B9563F/sc-minidriver_specs_v7.07.docx

A Appendix - použití

OAuth 2.0:
Před načtením PKCS#11 modulu je zapotřebí získat přístupový token k systému Remsig. Aplikace Oauth vytvoří složku, do které po úspěšné autentizaci uloží tyto přistopové údaje. Jestliže není soubor, či složka k dispozici, anebo není přístupový token platný, PKCS#11 modul se nepodaří načíst.

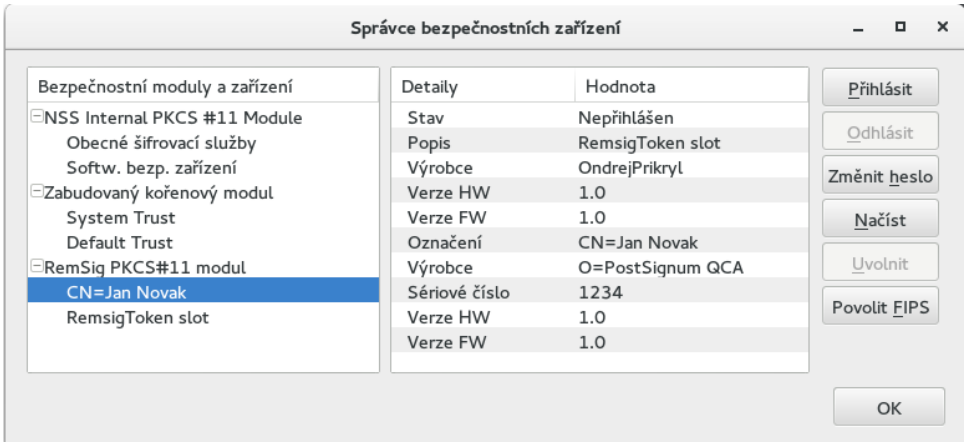
Přidání PKCS#11 modulu do poštovního klienta Mozilla Thunder-Bird:

1. Uložte si PKCS#11 modul na pernamentní místo na lokálním disku.

2. Po spuštění klienta otevřete okno nastavení. Zvolte "Zabezpečení", dále "Bezpečnostní zařízení".

3. Ve správci bezpečnostních zařízení zvolte možnost "Načíst".

4. Nyní pomocí možnosti "Procházet"zvolte cestu k *.so* souboru modulu a potvrďte tlačítkem "Ok".



Obrázek A.1: RemSig PKCS11 modul.