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.

Shrnutí

«abstract»

Klíčová slova

remsig, pkcs11, csp, oauth2.0, ...

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) [1] , 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 [2] a IS MU [3] , 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.

V první fázi práce se zaměřuji na vytvoření knihovny v programovacím jazyce C. Tato knihovna obsahuje základní funkce pro práci s úložištěm certifikátů v operačním systému Windows pomocí vývo-

1. https://cs.wikipedia.org/wiki/Elektronický_podpis.
2. `https://inet.muni.cz`
3. `https://is.muni.cz`

jového prostředí Cryptography API: Next Generation3 (dále CNG). V druhé fázi je zapotřebí vytvořit knihovnu poskytující komunikaci mezi klientem a serverem a pomocí otevřeného protokolu OAuth 2.0 získat přístup k datům uloženým na serveru RemSig. V poslední fázi spojuji již implementované knihovny do fungujícího celku.

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. Dalším krokem při vývoji systému RemSig by mohla být integrace jeho funkcí do systémů iOS a Linux. Podpora těchto tří operačních systémů by RemSigu umožnila fungovat na většině desktopových počítačů na světě.

2 Digitální podpis

Digitální podpis slouží k nahrazení obyčejného podpisu v digitálním světě. Je založen na asymetrické kryptografii a k jeho použití je zapotřeba dvojice klíčů - soukromého a veřejného. Digitální podpis vyžaduje použití

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.

To bez digitálního podpisu není tak samozřejmé, jak by se mohlo zdát. Digitální podpis je tvořen tzv. certifikátem, který je vydáván certifikační autoritou. Integrita je zajištěna pomocí hašovacích funkcí

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.

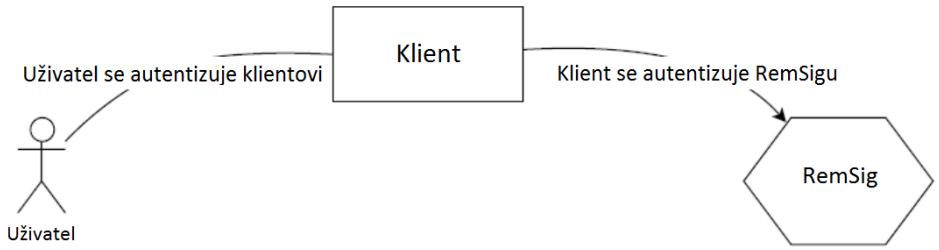
3.2 Autentizace

Pro přístup k API systému je zapotřebí, aby se uživatel autentizoval klientovi, který zprostředkovává komunikaci mezi uživatelem a veřejně přístupnou API RemSigu.

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 pří-

1. Kvalifikovaná certifikační autorita České Republiky, viz. `www.postsignum.cz/`



Obrázek 3.1: RemSig - Autentizace pomocí klienta.

stup 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.

3.3 Autorizace

Poté, co se klient autentizoval, Remsig obdrží 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. RemSig rozlišuje dva typy rolí:

- podpisující
- manažer

Podpisující má přístup k následujícím operacím:

- *sign* - podepíše data podle defaultních kryptografických mechanismů

2. viz. `https://cs.wikipedia.org/wiki/Access_control_list`

- *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 Správa úložiště

Aby uživatel mohl používat systém RemSig, musí v něm mít uložen certifikát a soukromý klíč. Remsig umožňuje generování této dvojice přímo v systému. Po vygenerování je soukromý klíč zašifrován pomocí přiloženého PINu, který si uživatel sám zvolí. Dvojice je poté uložena do úložiště a uživateli je nazpět zaslán certifikát, který je nutný 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.

---

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`

3.4.1 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 klíč- certifikát vygenerována systémem RemSig a certifikát byl předán na podepsání certifikační autoritou.

3.4.2 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.

3.4.3 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.

3.5 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.

3.5.1 Defaultní podpis

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

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 na serveru. 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.

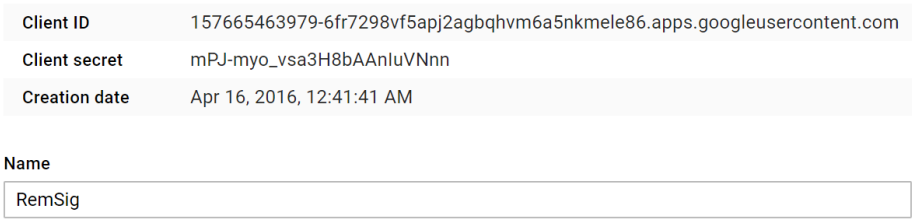
- Zasláním přihlašovacích údajů:
  V tomto typu uživatel poskytne přihlašovací údaje klientovi. Klient zašle přihlašovací údaje autorizačnímu serveru, který po úspěšné autentizaci uživatele zašle zpět přístupový token. Přihlašovací údaje jsou použity jen pro získání přístupového a obnovujícího tokenu, 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...).

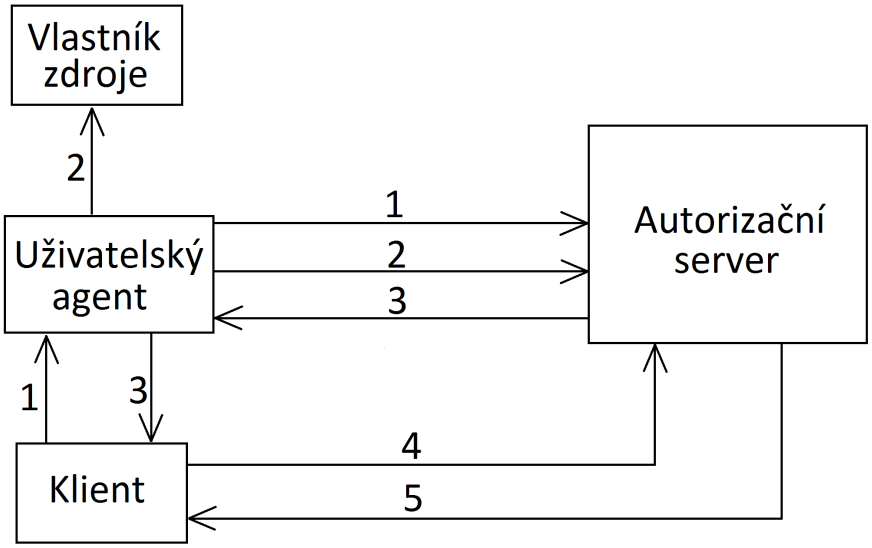
Poté co se klient úspěšně zaregistruje, obdrží od autorizačního serveru unikátní identifikátor a heslo. Tyto údaje jsou nutné pro autentizaci klienta serveru během průběhu protokolu. Je-li při průběhu vyžadována adresa přesměrování, tak musí souhlasit s adresou uvedenou klientem při registraci.



Obrázek 4.1: Google OAuth - registrace klienta.

4.7 Praktická část

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



Obrázek 4.2: 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.

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 ještě nebyl standardizován.

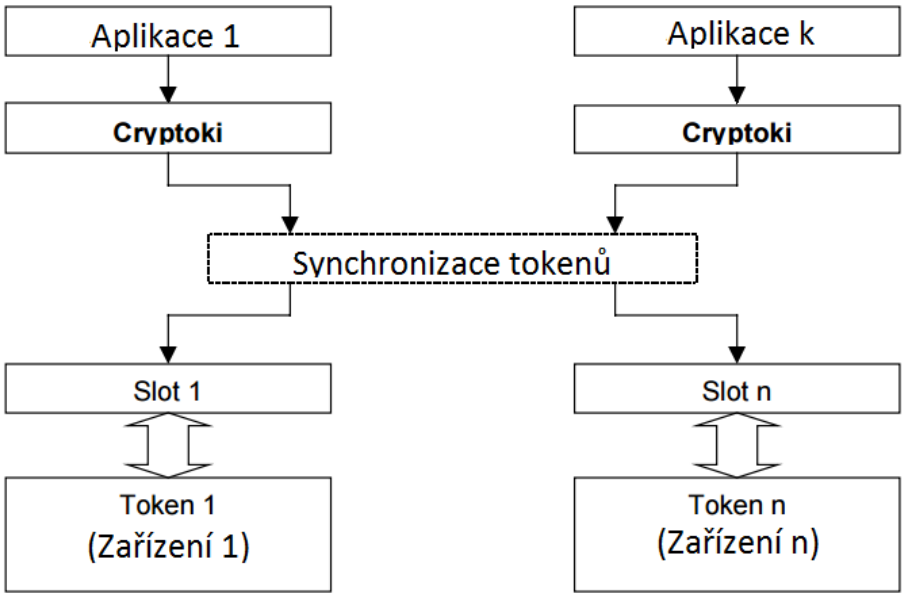
5.1 Cíle návrhu

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, PCMCIA 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.

5.2 Obecný model

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.

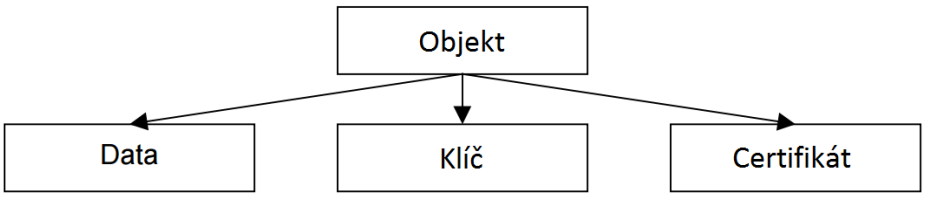
Díky tomu, že Cryptoki maskuje skutečnou hardwarovou podobu připojeného zařízení a zobrazuje je jako totožné jednotky, volající aplikace nemusí znát přesné rozhraní ke kterému je zařízení připojeno či jaký ovladač pro ně použít.

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

sdfsd

5.3 Praktická část