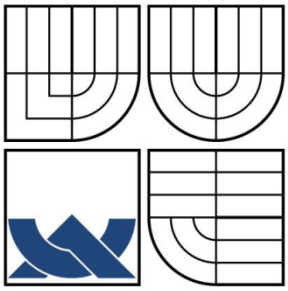
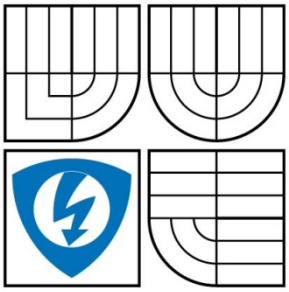


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO ELEKTRONICKÝ PODPIS A ČASOVÉ RAZÍTKO

APPLICATION FOR DIGITAL SIGNATURE AND TIMESTAMP

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

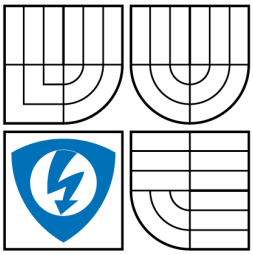
AUTOR PRÁCE
AUTHOR

BC. MIROSLAV REMIAŠ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PETRA LAMBERTOVÁ

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Miroslav Remiaš	***ID:***	83302
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Aplikace pro elektronický podpis a časové razítko

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a implementujte webovou aplikaci postavenou na platformě .NET Framework, která bude představovat certifikační autoritu. Aplikace bude schopna generovat certifikáty, digitálně podepisovat dokumenty a následně ověřovat pravost digitálního podpisu, umožní taktéž uživatelům bezpečně komunikovat pomocí e-mailových zpráv. Při podepisování a ověřování dokumentů se aplikace bude dotazovat autority časových razítek, která odpoví zasláním časového razítka dokumentu.

DOPORUČENÁ LITERATURA:

[1] Elektronický podpis : Přehled právní úpravy, komentář k prováděcí vyhlášce k zákonu o elektronickém podpisu a výklad základních pojmů. Olomouc : ANAG, 2002. 141 s. ISBN 80-7263-125-X.
[2] DOSTÁLEK, Libor, VOHNOUTOVÁ, Marta. Velký průvodce infrastrukturou PKI a technologií elektronického podpisu. 1. vyd. Brno : Computer Press, 2006. 534 s. ISBN 80-251-0828-7.
[3] DOSTÁLEK, Libor. Velký průvodce protokoly TCP/IP : Bezpečnost. 2. aktualiz. vyd. Praha : Computer Press, 2003. xvi, 571 s. ISBN 80-7226-849-X.

Termín zadání:	9.2.2009	***Termín odevzdání:***	26.5.2009

Vedoucí práce: Ing. Petra Lambertová

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstract

In general, the Internet represents an unsecured medium of data transfer. Besides the rising popularity of the Internet, the matters of safety are getting to the foreground of importance. Anybody would be able to gain access to the computer network or to other valuable information if no algorithm of verifying the genuineness of identity were used. It is necessary to secure not only the access to the documents but also the content itself, which could be modified during the transfer through an unsecured medium. Last but not least, without the discretion provided by cryptography, the information may become literally public. To provide security and protection for the communicating participants the problems mentioned above are solved with the help of cryptographic techniques. The verification of the identity and the integrity of messages, the credibility of document's ownership and safe data transfer through an unsecured medium are all the aspects, which the field of communication security on the Internet, thus the public key infrastructure, deals with. The electronic signature, as a part of the security area, is one of many advertised themes nowadays in Czech Republic.

The aim of this master's thesis is to acquaint the reader with the necessary technological procedures of digital signature, such as cryptographic techniques, public key infrastructure and timestamp. The practical part of this thesis consists of a suggested implementation of a web application in the programming language ASP.NET, which forms a certification authority with an opportunity of claiming a timestamp to authorize timestamps.

After the problematic of cryptography was explained in the first chapter, the term of electronic signature has been introduced in the second chapter. Very important information, as far as the electronic signature of documents is concerned, is the time of the document's creation and the subsequent signature verification by an appropriate authority. So the following part of the thesis is dedicated to the timestamp and to the authority of its verification. The fourth section deals with the large scale of public key infrastructure. The fifth part focuses on the description of the support for the whole problem mentioned so far using Microsoft's programming language ASP.NET.

The final sixth chapter represents the practical part of the thesis, namely the web application itself, where the individual modules of the application with its functions are described.

Keywords: TSA, CA, PKI, cryptography, key, digital signature, timestamp, authority

Anotace

Internet všeobecně poskytuje nezabezpečené přenosové médium. Se zvýšenou popularitou internetu se dostávají do popředí i otázky týkající se bezpečnostní problematiky. Bez algoritmů ověřování pravosti identity by si kdokoliv mohl zajistit přístup do počítačové sítě nebo k jiným cenným informacím. Na rozdíl od přístupu k informacím, je nevyhnutelné zajistit i samotný obsah dokumentů, které by mohly být v průběhu přenosu skrz nezabezpečené médium modifikované. V neposlední řade bez šifrování, které poskytuje určitou diskrétnost, se informace může stát skutečně veřejnou. Problémy poskytnutí bezpečnosti a ochrany komunikujících účastníků řešíme pomocí kryptografických technik. Ověření identity, integrity zpráv, důvěryhodnosti vlastnictví dokumentů a bezpečný přenos informací skrz nezabezpečené médium, to všechno jsou aspekty, kterými se zaobírá oblast bezpečnosti internetové komunikace - infrastruktura veřejných klíčů. Problematika elektronického podpisu, která spadá do této oblasti, patří v současné době v České republice k velmi diskutovaným a medializovaným problematikám.

Cílem následující diplomové práce je stručně obeznámit čitatele s nutnými technologickými aspekty digitálního podpisu - kryptografickými technikami, infrastrukturou veřejných klíčů a časovým razítkem. Praktická část diplomové práce představuje navrhnutou a implementovanou webovou aplikaci v programovacím jazyku ASP.NET, která tvoří certifikační autoritu s možností dotazování se o časové razítko na autoritu časových razítek.

Na základe objasnění problematiky kryptografie v kapitole č.1 je následně zaveden pojem elektronický podpis a jeho souvislosti, které jsou prezentované v kapitole č.2. Důležitou informací při digitálním podpise dokumentů je časový údaj pořízení dokumentu a následné ověření tohoto podpisu příslušnou autoritou, a proto je v následující části věnovaná pozornost právě časovému razítku a autoritě k jeho ověření. Čtvrtá část této diplomové práce se věnuje rozsáhlé problematice infrastruktury veřejného klíče. V páté kapitole je soustředěn popis na podporu celé doposud popsané problematiky v programovacím jazyku od společnosti Microsoft - ASP.NET.

Závěrečná, šestá kapitola, představuje praktickou část diplomové práce, a to samotnou webovou aplikaci. Jsou zde popsány jednotlivé moduly aplikace a jejich funkcionalita.

Klíčová slova: TSA, CA, PKI, kryptografie, klíč, digitální podpis, časové razítko, autorita

Bibliografická citace práce:

REMIAŠ, M. *Aplikace pro elektronický podpis a časové razítko.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 45 s. Vedoucí diplomové práce Ing. Petra Lambertová.

Prohlášení

Prohlasuji, že svůji diplomovou práci na téma Aplikace pro elektronický podpis a časové razítko jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou vedeny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Petre Lambertovej, techn.hosp.pracovníkovi, za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne …… ……..

(podpis autora)

Obsah

Zoznam obrázkov

Úvod

V priebehu stáročí boli luďstvom vypracované rady metód pre overovanie obsahu a podpisu dokumentov, ktoré sa používali pre dokazovanie vlastníctva alebo pre komunikáciu vzdialených osôb. Tvorcovia týchto prostriedkov však vedeli, že aj základné záznamové médium ľudstva (papier) je ľahko sfalšovateľné a že podpis je možno ľahko napodobniť, dokonca modifikovať obsah už podpísaného dokumentu. Postupom času a hlavne s príchodom internetu, ako globálneho komunikačného prostriedku, sa začali prenášať tieto problémy aj do moderných spôsobov elektronickej komunikácie.

Internet je verejne dostupný celosvetový systém vzájomne prepojených počítačových sietí, cez ktoré sa prenášajú dáta. Pozostáva z tisícky menších komerčných, akademických, vládnych a vojenských sietí. Je prenosovým médiom rôznych druhov informácií a služieb, akými sú napríklad elektronická pošta, elektronické bankovníctvo a mnoho ďalších. Internet poskytuje užívateľom anonymitu, ktorá má za úlohu zabezpečovať súkromie na jednej strane, na druhej strane je však táto veľmi dobrým prostredím pre nelegálnu činnosť. Kriminalita, ako jeden z najväčších problémov ľudstva, sa odjakživa vyskytovala v každej oblasti ľudskej činnosti a ani internet sa jej taktiež nevyhol. S rastom počtu užívateľov internetu analogicky rastie aj počet osôb, ktoré vykonávajú rozličné ilegálne aktivity a keďže internet je nezabezpečená zóna, kde sa teoreticky ktokoľvek môže nachádzať medzi komunikujúcimi účastníkmi prevádzajúcimi rôzne diskrétne operácie, sú užívatelia, rovnako ako aj poskytovatelia internetu postavení pred problém bezpečnosti. Problémy, ktoré sa zaoberajú autentizáciou, overovaním komunikujúcich osôb, problémy spočívajúce v zaistení integrity správ a mnoho ďalších sú riešené v oblasti infraštruktúry verejných kľúčov. Tieto problémy sú zamerané hlavne na overovanie digitálneho podpisu komunikujúcich strán a na podmienku nepopierateľnosti, kedy je prenášaná správa vytvorená práve jedným z komunikujúcich osôb. A práve kryptológia, ako vedný odbor, ktorý sa zaoberá šifrovacími a kódovacími algoritmami, rieši tieto problémy a stáva sa základným a nevyhnutným prostriedkom pre fungovanie celej infraštruktúry verejných kľúčov. V súčasnej dobe sú najviac diskutované témy autentizácie komunikujúcich strán a s tým súvisiaci elektronický a digitálny podpis, ktorý vo svojej podstate zahrňuje najmodernejšie kryptografické metódy, kombinujúce symetrické a asymetrické šifrovacie a kódovacie algoritmy. Keďže samotný digitálny podpis sám o sebe nerieši problematiku časového obdobia, v ktorom tento podpis vznikol, je potreba túto situáciu riešit za pomoci pridania časového údaja k digitálnemu podpisu - časového razítka.

V diplomovej práci Aplikace pro elektronický podpis a časové razítko si kladiem za cieľ zmapovať a vysvetliť problematiku infraštruktúry verených kľúčov a s ňou súvisiacimi témami a vytvoriť praktickú ukážku webovej aplikácie predstavujúcej certifikačnú autoritu pomocou jazyka ASP.NET.

Na začiatku, v prvej kapitole, budem venovať pozornosť kryptológií ako nutnému základu pre pochopenie celej tematiky. V ďalšej kapitole sa zameriavam na digitálny podpis a jeho princíp. Dôležitým údajom pri digitálnom podpise je časový údaj vytvorenia podpisovaného dokumentu, ktorý je označovaný ako časové razítko. Práve vo štvrtej kapitole sa zameriavam na opis časového razítka a autorite na jeho overovanie a vydávanie. V dalšej časti popisujem programovací jazyk ASP.NET, ktorý sa v dnešnej dobe s obľubou používa pre tvorbu dynamických stránok a je vhodný napríklad pre tvorbu aplikácií postavených na princípe elektronického bankovníctva. V poslednej kapitole popisujem vytvorenú webovú aplikáciu, ktorá prakticky prezentuje nadobudnuté teoretické poznatky zo všetkých predchádzajúcich kapitol.

1 Úvod do kryptografie

Vedu o využívaní matematických funkcií pre kódovanie a opätovné získanie dát, ktorá sa taktiež zameriava na návrh šifrovacích algoritmov môžeme označiť ako kryptografiu. Uchovávanie tajných informácii, zaisťovanie dôvernosti chránených dát a ich distribúcia po nezabezpečenom médiu sú hlavnými úlohami tohto odvetvia bezpečnosti. Správa pred zašifrovaním (otvorený text) je podľa predom dohodnutých pravidiel odosielateľa a príjemcu pozmenená na šifrovaný text, aby nedošlo k zneužitiu jej obsahu. V prípade získania takto zmodifikovanej správy útočníkom je jej odhalenie bez znalosti presných pravidiel pre dešifrovanie veľmi obtiažne, v závislosti na použitej technológii vo väčšine prípadov však nemožné.

Dáta ktoré sú chránené kryptografickými prostriedkami, by nemal mať možnosť nikto prečítať ani pri nasadení najmodernejších výpočtových systémov pre ich vylúštenie. Bezpečnosť množstva algoritmov v minulosti bola založená na ich dokonalom utajení avšak problém nastal práve vtedy, ak sa prezradil princíp činnosti algoritmu. Z tohto dôvodu sa v súčasnosti presadzuje trend zverejňovania všetkých kryptografických algoritmov. Bezpečnosť algoritmov je teda založená na predpoklade matematickej náročnosti riešiť úlohy v reálnom čase a nie na dôslednom utajení. Taktiež zverejnený algoritmus pôsobí dôveryhodnejšie a každý má možnosť skontrolovať jeho kvalitu.

Hlavnými problémami ktorými sa kryptografia zaoberá, môžeme rozdeliť do nasledujúcich bodov:

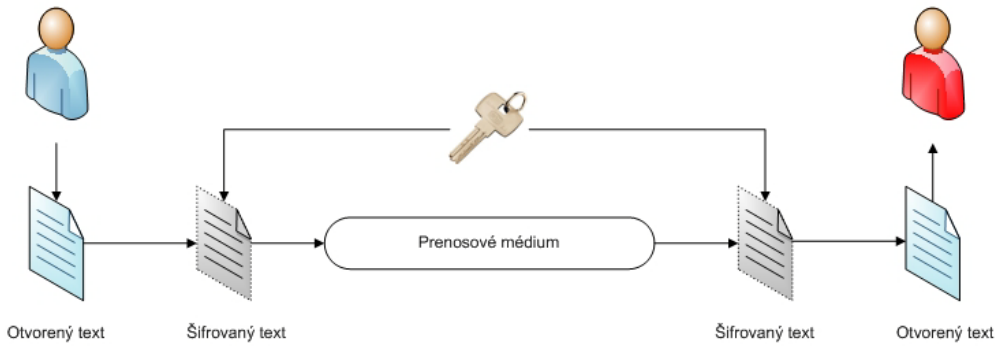
a) Návrh a analýza kryptosystémov
b) Šifrovanie/dešifrovanie dát
c) Integrita a autentičnosť dát
d) Overenie identity komunikujúcich strán

Kryptografiu je možné rozdeliť z hľadiska spôsobu šifrovania na:

a) symetrickú.
b) asymetrickú.

1.1 Symetrická kryptografia

Princíp činnosti algoritmu symetrickej kryptografie spočíva vo využití jedného kryptografického kľúča, ktorý je spoločný pre obe komunikujúce strany (viz obr.1.1). Identický kľuč sa používa preto ako pre šifrovanie tak i pre dešifrovanie dát. Hlavnou výhodou tohto princípu komunikácie je jeho nízka výpočtová náročnosť a s tým súvisiaca vysoká prenosová rýchlosť v porovnaní s asymetrickými algoritmami. Na druhú stranu tu vzniká nevýhoda v podobe vyšších nárokov na počet kľúčov a ich spravovanie. Medzi najzákladnejšie techniky, ktoré využívajú symetrické algoritmy patria substitúcia a transpozícia. Prvá z nich nahradzuje znak otvoreného textu znakom šifrovaného textu podľa predpísaného kľúča. Transpozícia zachováva hodnotu znakov, ale mení ich poradie. Množstvo algoritmov využíva kombináciu týchto dvoch metód [4].



Obr.1.1: Schéma symetrickej kryptografie.

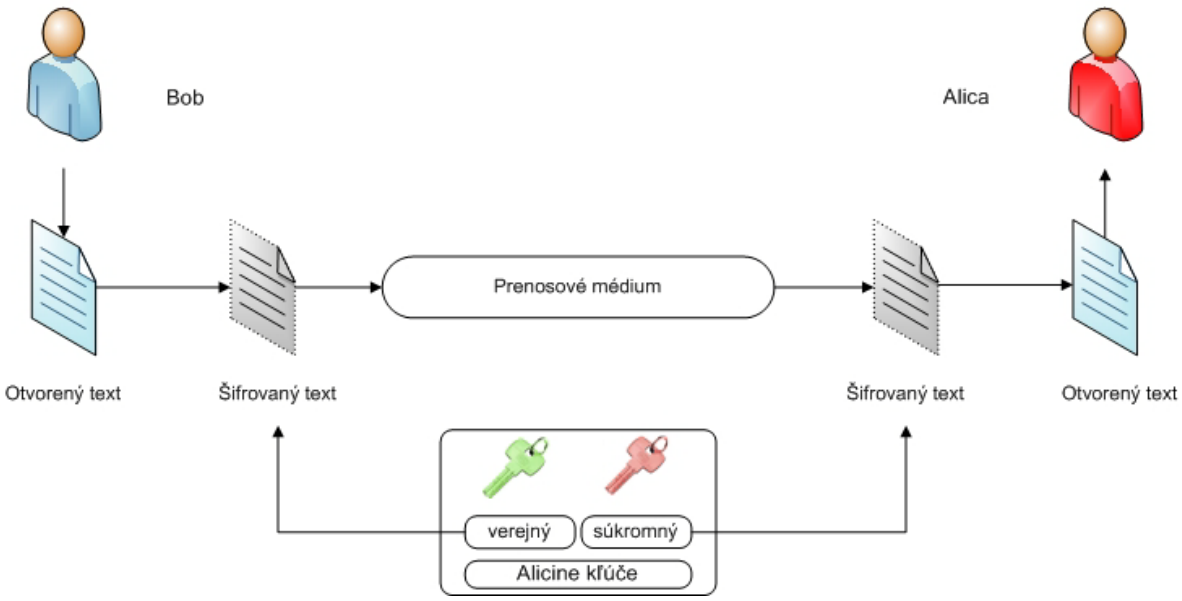
1.2 Asymetrická kryptografia

Použitie dvoch kľúčov pre každého z komunikujúcich účastníkov je základným princípom asymetrickej kryptografie. Jedná sa o jeden pár kľúčov, ktorý pozostáva z verejného a súkromného kľúča. Najpodstatnejšou vlastnosťou týchto kľúčov je, že dáta zašifrované jedným kľúčom z tejto dvojice môžu byť dešifrované iba a práve tým druhým kľúčom z toho páru. Obmedzenie platí aj pre šifrovanie a následné dešifrovanie identickým kľúčom, kedy zašifrované dáta jedným kľúčom nie je možné týmto identickým kľúčom spätne dešifrovať.

Bezpečnosť asymetrickej kryptografie spočíva v neprístupnosti súkromného kľúča a jeho bezpečného utajenia, oproti tomu sa verejný kľuč poskytuje všetkým komunikujúcim stranám. Pokiaľ užívateľ chce komunikovať, musí poskytnúť svoj verejný kľúč. Týmto kľúčom mu príde zašifrovaná správa od komunikujúceho na druhej strane. Jediný, kto môže túto správu prečítať alebo dešifrovať je užívateľ vlastniaci súkromný kľuč z páru kľúčov, z ktorého bol poskytnutý aj verejný kľuč. Systém tohto šifrovania sa zakladá na matematických riešeniach, ktoré neprinášajú žiadne výsledky v polynominálnom ale iba v exponenciálnom čase. Asymetrické šifry väčšinou pracujú so špecifickým druhom čísiel - s prvočíslami. V dnešnej dobe sa pracuje prevažne s 1024 a 2048 bitovými kľúčmi. Výhodou tohto princípu komunikácie je potreba oveľa menšieho počtu kľúčov, kedy každá osoba vlastní iba jeden pár kľúčov. Cenu za vyššiu bezpečnosť v porovnaní so symetrickou kryptografiou si na druhú stranu vyžiadala vyššia výpočtová náročnosť algoritmov a tým pádom aj pomalšia komunikácia [4].

Príklad komunikácie:

Bob chce poslať správu Alici. Správu zašifruje Aliciným verejným kľúčom a odošle. Na druhej strane Alica príjme túto správu a dešifruje ju svojím súkromným kľúčom (viz obr.1.2).

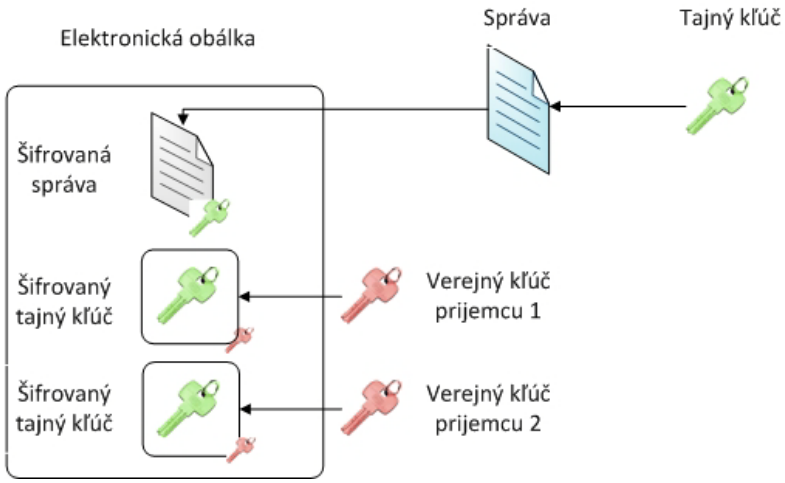


Obr.1.2: Schéma asymetrickej kryptografie.

1.3 Elektronická obálka

V prípade použitia asymetrickej kryprografie by bolo zbytočne výpočtovo náročné používať tieto matematické postupy pre samotné šifrovanie dát, kde je doba trvania výpočtu veľmi dlhá. Jedným z riešení tohto problému je použitie elektronickej obálky, kedy užívateľ zašifruje správu súkromným kľúčom. Táto operácia je pomerne rýchla. K takejto správe pridá informáciu pre

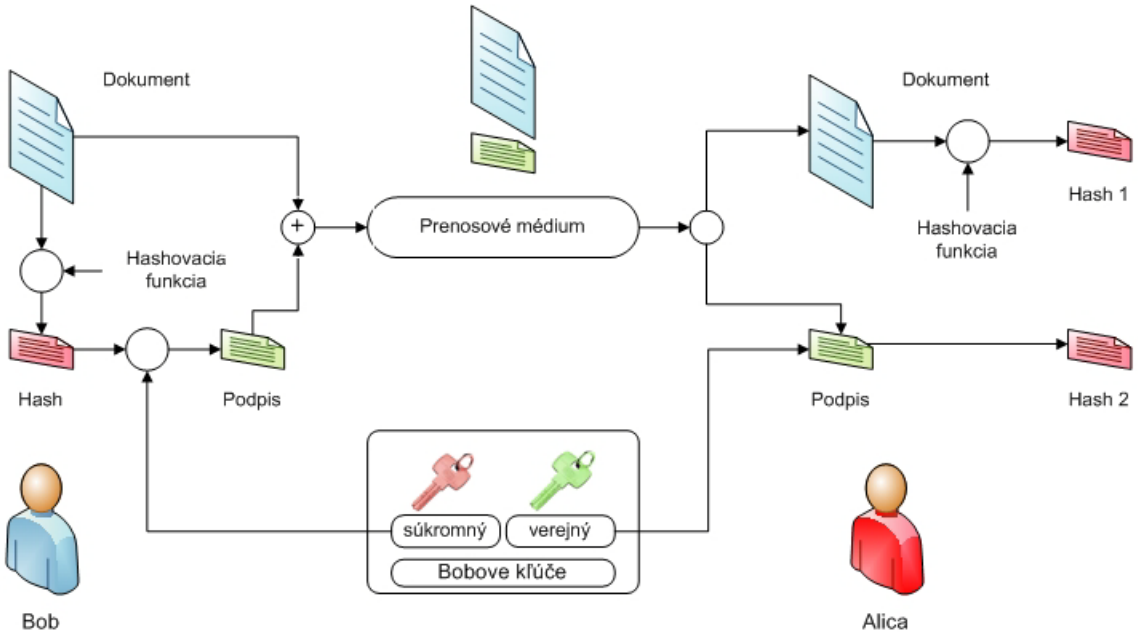
príjemcu, ktorá bude obsahovať tento symetrický kľúč pre dešifrovanie v nečitateľnej podobe, pretože bude šifrovaný verejným kľúčom príjemcu (viz obr.1.3). Tu nastáva výhoda rýchlosti prevedenia tejto operácie, pretože sa asymetricky šifruje iba tajný (symetrický) kľúč. Ďalšiu výhodu tejto techniky je možné nájsť pri potrebe posielania tejto zašifrovanej správy viacerým prijímateľom, pretože správa sa šifruje iba raz a každému adresátovi v tejto celej správe poskytneme tajný kľúč zašifrovaný jeho verejným kľúčom [2].



Obr.1.3: Elektronická obálka.

2 Digitálny podpis

Pri komunikácii užívateľov na veľké vzdialenosti a potrebe overia identity týchto komunikujúcich účastníkov, či už pri podpisoch rôznych zmlúv alebo bankových transakciách, sú nútený dokázať svoju identitu nie pomocou klasického osobného podpisu, ale pomocou digitálneho podpisu. Zaistenie nepopierateľnosti, čiže jednoznačnú identifikáciu užívateľa je úlohou digitálneho podpisu. Ďalším nutným aspektom pri správnej komunikácii je zaistenie integrity podpísaného textu. To znamená byť schopný odhaliť, či bol daný text nejakým spôsobom pozmenený, pretože na komunikačnom médiu môže teoreticky ktokoľvek túto správu pozmeniť. Na rozdiel od princípu asymetrickej kryptografie je digitálny podpis vytváraný pomocou súkromného kľúča asymetrického kryptografického systému a hashovacej funkcie (viz kapitola 2.3). Jeho správnosť je overovaná naopak verejným kľúčom, ktorý tvorí spolu so súkromným kľúčom taktiež pár kľúčov. Dôvodom tohto rozdielu je odlišná potreba, pretože pri podpise dokumentu chceme aby si ktokoľvek mohol overiť platný podpis. Platný podpis nesmie byť schopný vytvoriť nikto okrem oprávneného človeka [2].



Obr.2.1: Schéma princípu digitálneho podpisu.

Vytvorenie digitálneho podpisu dokumentu:

1.) Vytvoríme hash podpisovaného dokumentu pomocou jednosmernej funkcie. Tento charakterizuje obsah dokumentu a mal by byť pre každý dokument jedinečný.
2.) Hash dokumentu zašifrujeme s použitím súkromného kľúča a týmto spôsobom získame digitálny podpis. Takýto podpis sa nedá podvrhnúť, pretože nikto nemá prístup k súkromnému kľúču, iba je ho vlastník.
3.) Dokument spolu s jeho podpisom odošleme príjemcovi.

Overenie digitálneho podpisu dokumentu:

1.) Pomocou totožnej jednosmernej funkcie ako bola použitá v prípade tvorby digitálneho podpisu vytvoríme hash1 dokumentu, ktorého pravosť chceme overiť. Informácia o použitej jednosmernej funkcii je súčasťou digitálneho podpisu.
2.) S použitím verejného kľúča odosielateľa, autora podpisu, dešifrujeme podpis, získame hodnotu hash2.

3.) Porovnáme hodnoty hash1 a hash2, ak sú rovnaké ide skutočne o ten istý dokument, ktorý bol podpísaný vlastníkom súkromného kľúča v nezmenenej podobe.

2.1 Elektronický a zaručený elektronický podpis

Doposiaľ bol spomínaný iba digitálny podpis, pretože tento termín použitý v celej práci a môžeme ho chápať ako podpis vytvorený na základe asymetrickej kryptografie, tak ako je popisovaný v tejto kapitole. Existuje však aj elektronický podpis, ktorý je ľahšie pochopiteľný pre právnikov, vhodnejší pre legislatívu a môžeme ho zase chápať ako všetky elektronicky vytvorené dôkazy o tom, že dokument bol vytvorený a podpísaný určitou osobou.

Kedže digitálny podpis môže slúžiť aj ako dôkaz pravosti dokumentu, môže byť za určitých podmienok braný ako plnohodnotná náhrada rukou písaného podpisu. Takýto podpis je potom označovaný ako zaručený elektronický podpis. Vytvorenie zaručeného elektronického podpisu podlieha nie len kryptografickým parametrom ale aj legislatívnym podmienkam štátu, kde sa má daný zaručený podpis používať.

V niektorých štátoch je elektronický podpis chápaný ako plnohodnotná náhrada rukou písaného podpisu. V týchto štátoch je potom v právnom rade zavedený výraz zaručený elektronický podpis. V iných štátoch je zase digitálny podpis chápaný výhradne iba k autentizácii dokumentov ale nie je braný ako plnohodnotná náhrada rukou písaného podpisu.

Zaručeným elektronickým podpisom, ktorý je zapracovaný do českej legislatívy v zákone "O elektronickom podpise" č. 227/2000 Sb., ktorý bol neskôr novelizovaný zákonmi 226/2002 Sb., 517/2002 Sb. a 440/2004 Sb., sa myslí elektronický podpis, ktorý splňuje nasledujúce požiadavky [2]:

- *Je jednoznačne spojený s podpisujúcou osobou.*
- *Umožňuje jednoznačne identifikovať podpisujúcu osobu vo vzťahu k dátovej správe.*
- *Bol vytvorený pomocou prostriedkov, ktoré podpisujúca osoba môže udržať pod svojou kontrolou.*
- *Je k dátovej správe pripojený takým spôsobom, že je možné zistiť akúkoľvek následnu zmenu dátovej správy.*

2.2 Zaistenie integrity dát

Pri požiadavke na bezchybný prenos dát, či už kvôli modifikácii správy útočníkom počas prenosu alebo strate, potrebujeme tieto dáta na strane príjemcu nejakým spôsobom overiť. Preto je nevyhnutné vytvoriť na strane odosielateľa jedinečný kontrolný súčet, ktorý by pri kontrole správy na strane príjemcu dával rovnaký výsledok. Pre špeciálnu skupinu matematických funkcií, ktorým sa hovorí jednosmerné, je veľmi jednoduché vypočítať hodnotu funkcie pri zadaných vstupoch, naopak je v konečnom čase prakticky nemožné stanoviť z výsledku funkcie pôvodný vstup.

2.3 Hashovacie funkcie

Jednosmerné hashovacie funkcie sú tiež nazývané Message Digest algoritmy. Pri potrebe poslať zašifrovanú správu takým spôsobom, aby jej obsah bol dostupný pre čítanie konkrétnemu adresátovi, predstavuje táto situácia riziko bez použitia hashovacej funkcie. Počas cesty tejto správy by mohol niekto modifikovať obsah správy a príjemcovi by prišla správa modifikovaná, z čoho vyplýva nebezpečenstvo pozmenenia správy. Práve k tomuto účelu sa používa špeciálny kontrolný sumarizačný kód správy, ktorý je doručený príjemcovi spolu s originálnou správou. Príjemca si znovu spraví sumarizáciu správy a porovná prijatý kód s kódom, ktorý si vygeneroval z prijatej správy. V prípade, že kódy sú totožné, znamená to, že správa nebola modifikovaná a jej

obsah je rovnaký ako ho poslal odosielateľ. Ak kódy nie sú totožné je správa buď poškodená, alebo modifikovaná. Takáto sumarizácia sa nazýva Message Digest (MD) alebo hash. MD algoritmus bol navrhnutý tak, aby generoval unikátnu sumarizačnú značku pre rozdielne správy. Je založený na princípe, kedy sa z neho spätne nedá získať správa, pomocou ktorej bol hash vytvorený. Rovnako je ťažké nájsť dve rôzne správy, z ktorých by bola vygenerovaná rovnaká sumarizácia, aj keď je to teoreticky možné [4].

3 Časové razítko

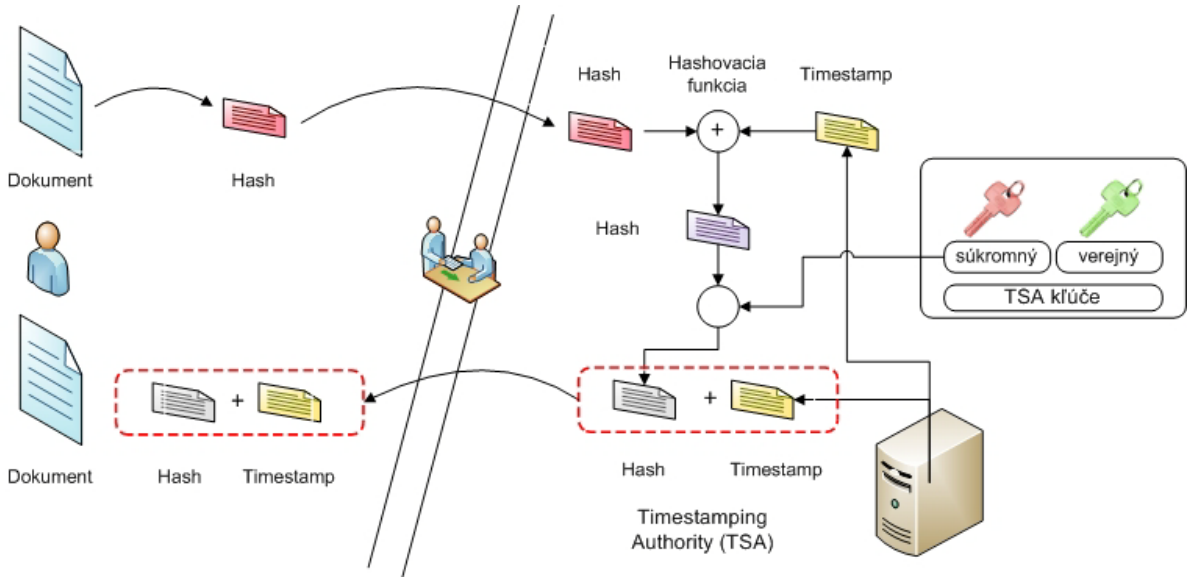
V prípade podpisu nejakej zmluvy sa text vo väčšine prípadov ukončuje podpisom, dátumom a eventuálne miestom kde bol dokument podpísaný. Pokiaľ by v dokumente nebol uvedený dátum podpisu, bol by tento dokument určite sporný. Požadovanou a neoddeliteľnou súčasťou každého písomného dokumentu je teda podpis a dátum jeho vzniku, či doba platnosti. Problém môže nastať pri dokumentoch, ktoré majú dlhú dobu platnosti a je potreba ich overiť aj s určitým časovým odstupom. Keďže v mnohých prípadoch pôvodné certifikáty už stratili svoju platnosť, nie je možné daný dokument regulérne overiť z dôvodu neexistencie právoplatných údajov o čase vzniku dokumentu.

Technika, ktorá má v tomto prípade zastúpenie v oblasti overenia časovej platnosti podpisu sa nazýva časové razítko (TS - Time Stamp). V súvislosti s elektronickým podpisom môže byť teda dôležitá informácia, či bol dokument elektronicky podpísaný v dobe platnosti certifikátu, na ktorom je vlastne elektronický podpis založený, resp. či bol dokument podpísaný skôr ako bol certifikát zrušený.

Tento dokument je teda orazítkovaný časovým razítkom na vyžiadanie takzvanou Autoritou časových razítok (Time Stamping Authority - TSA). Nutným základom pre poskytovanie elektronických služieb a zaistenie dlhodobého overenia elektronicky podpísaných dokumentov je teda Autorita časových razítok [2].

Súčasťou požiadavku o vytvorenie časového razítka môžu byť obecne dáta, pre ktoré chceme získať časové razítko. Toto obsahuje aktuálny dátum, čas, sériové číslo razítka a identifikáciu TSA autority. Podrobnosti k tejto špecifikácii je možné nájsť v dokumente RFC 3161 [6].

Tieto údaje sa ďalej pripoja k vstupným dátam od žiadateľa a celok sa podpíše súkromným kľúčom TSA. Výsledok sa potom pošle žiadateľovi ako odpoveď na jeho žiadosť, ako je možné vidieť na obrázku 3.1. Pre podpisovaný dokument môžeme získať časové razítko, keď do žiadosti pre TSA uvedieme ako vstupné dáta hash daného dokumentu. Správa sa teda neposiela v tvare v akom vznikla, ale posiela sa iba jej jednoznačná reprezentácia – hash. Príslušná autorita teda nie je závislá na obsahu razítkovaného dokumentu a nemá ani možnosť sa s ním zoznámiť. Potom práve dokument, elektronický podpis a časové razítko spolu zväzujú dokopy osobu, ktorá podpísala dokument ale aj časový moment, pred ktorým dokument zaručene existoval.



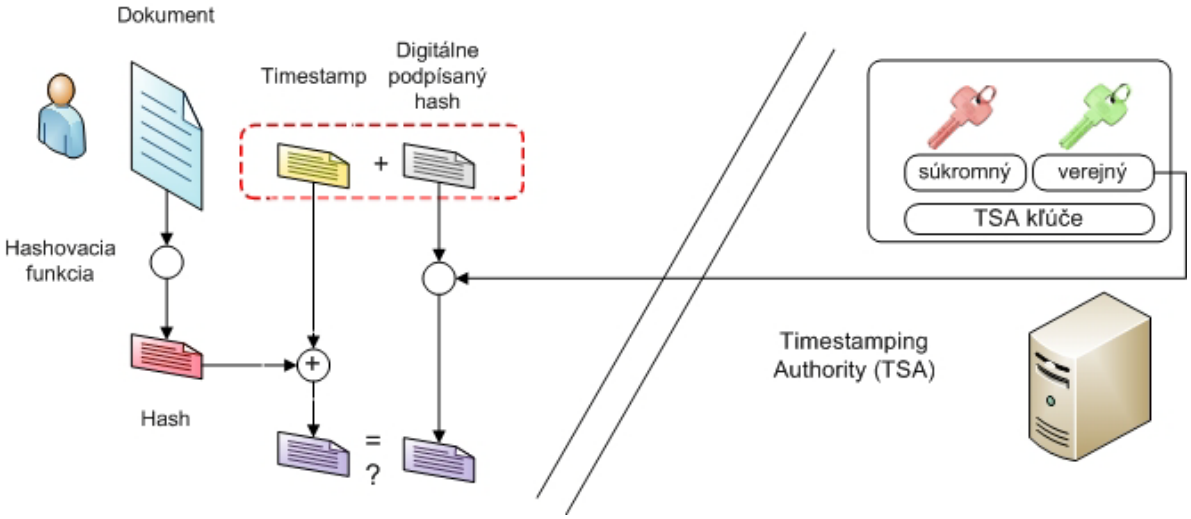
Obr.3.1: Schéma princípu vytvorenia časového razítka.

Dokument je v časovom razítku predstavuje tzv. *message imprint*. Tento pozostáva z algoritmu pre výpočet hashu, ktorý bol použitý k tvorbe hashu a samotného hashu dokumentu. Ako bolo už uvedené TSA neskúma a ani nesmie skúmať totožnosť žiadateľa časového razítka a preto časové razítko ani neobsahuje žiadny údaj, ktorý by identifikoval žiadateľa. Časové razítko je možné použiť v praxi pri razítkovaní dokumentov, kedy vznikne dvojica dokument a časové razítko. V tomto prípade dostávame indíciu o existencii dokumentu v čase. Každá ďalšia zmena dokumentu by totižto znamenala zneplatnenie časového razítka.

Ako druhá varianta použitia časového razítka je razítkovanie podpisu dokumentu. Príklad by sme mohli nájsť v digitálnom podpise bankovej transakcie. Banky väčšinou archivujú tieto transakcie po dlhšiu dobu ako je platnosť samotného certifikátu, ktorý je nutný pre overenie, kto danú transakciu prevádzal. Odosielateľ by mohol prehlásiť, že danú operáciu nevykonal. Môže tak argumentovať z toho dôvodu, že existuje predpoklad, že banka mala k dispozícii jeho verejný kľúč po dostatočne dlhú dobu na to, aby našla k nemu súkromný kľúč, s ktorým je potom schopná podpísať falošný platobný príkaz. Pokiaľ si ale táto banka nechá vytvoriť k danému podpisu platobného príkazu časové razítko, predíde sa podobnej špekulácii. Práve tento prístup je implementovaný v praktickej časti diplomovej práce.

Pre lepšiu pochopiteľnosť uvádzam príklad použitia tohto systému:

Predpoklad spočíva v podpísaní dlhodobej zmluvy, pre ktorú sme si zaobstarali časové razítko. Certifikát, s ktorým je zviazaná táto zmluva, je platný po dobu jedného roku. V tomto časovom období certifikačná autorita potvrdzuje platnosť elektronického podpisu pre danú osobu. Po uplynutí jedného roku platnosť certifikátu vyprší. Následné overenie správnosti digitálneho podpisu vyústi k záveru, že podpis síce odpovedá verejnému kľúču uvedenému v certifikáte, ale certifikát už nepotvrdzuje, že patrí subjektu, ktorý dokument podpísal. Preto stačí vyhodnotiť časové razítko. Ak spadá doba časového razítka do intervalu platnosti certifikátu elektronického podpisu, môžeme vyvodiť záver, že verejný kľúč patril v dobe platnosti certifikátu podpisujúcemu subjektu [2].



Obr.3.2: Schéma overenia pravosti časového razítka.

Z toho príkladu je zrejmé, že časové razítko teda neobsahuje identifikáciu subjektu a neslúži ako dôkaz o tom, že bezprostredne pred vydaním razítka mal dokument vo vlastníctve určitý subjekt. Predpokladom správneho fungovania systému časového razítka je použitie vhodných prostriedkov a postupov na strane poskytovateľa služby.

3.1 Dôveryhodný časový údaj TSA

Keďže jednou z najdôležitejších súčastí časového razítka je samotný časový údaj, nastáva otázka, či TSA dokáže obhájiť dôveryhodnosť časového údaju, ktorý používa vo svojich odpovediach na žiadosti o časové razítka. Primárnym zdrojom času by mal byť veľmi presný oscilátor, ktorý môže byť založený na prechode atómov vodíka, celzia a rubida. Druhým a nie menej dôležitým faktorom pri výbere dôveryhodného zdroju času je taktiež jeho cena. Autorita pre vydávanie časových razítok je vybavená zdrojom času s určitou odchýlkou, ktorá odpovedá politike tejto TSA. Vo väčšine prípadov používajú komerčné TSA rubidové oscilátory. V prípade menších TSA sa berie čas iba z jedného zaručeného zdroja. Naopak v prípade väčších TSA je doporučené brať čas aspoň z troch zdrojov [2].

3.2 TSA

Autorita časových razítok vytvára jeden alebo viacej dokumentov, ktoré tvoria politiku TSA. Každej politike sa potom priradí unikátny identifikátor objektu (OID – Object Indentifier), ktorý sa potom vkladá do časového razítka. TSA musí podpisovať časové razítka iba kľúčom, ktorý je výhradne určený k tomuto účelu. Verejný kľuč TSA je uložený v certifikáte TSA, ktorý musí obsahovať rozšírenie tohto certifikátu (Rozšírené použitie kľúča), ktoré musí byť označené ako závažné a obsahuje identifikátor objektu:

```
Id-kp-timeStamping OBJECT IDENTIFIER :: = {1.3.6.1.5.5.7.3.8}
```

Podrobné informácie o politikách pre TSA je možné nájsť v dokumente RFC-3628: Policy Requirements for Time-Stamping Authorities (TSAs) [7]. Správne navrhnutá TSA by mala taktiež dodržať nasledujúce body podľa špecifikácie RFC-3161: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP) [6]:

1.) *Použiť dôveryhodný zdroj času (špecifikovaný v politike TSA).*
2.) *Vkladať dôveryhodný časový údaj do každého vydaného časového razítka.*
3.) *Vkladať jedinečné sériové číslo (v rámci TSA) do každého vydaného časového razítka.*
4.) *Vydať časové razítko vždy ak je to možné, po obdržaní platnej žiadosti o časové razítko.*
5.) *Vkladať do každého vydaného časového razítka identifikátor politiky TSA, pod ktorou bolo razítko vydané.*
6.) *Algoritmus pre výpočet hashu musí byť jednoznačne identifikovaný svojim identifikátorom objektu (OID).*
7.) *Zkontrolovať, či dĺžka hashu odpovedá použitému algoritmu pre výpočet hashu.*
8.) *Nesnažiť sa žiadnym spôsobom analyzovať hash inak ako skontrolovať jeho dĺžku.*
9.) *Nevkladať do časového razítka žiadnu informáciu, ktorá by mohla identifikovať žiadateľa o časové razítko.*
10.) *K podpisu časového razítka použiť iba kľúče výhradne určené k tomuto účelu.*
11.) *V prípade ak žiadateľ použil v žiadosti o časové razítko také rozšírenie, ktoré TSA nepodporuje, TSA vráti chybové hlásenie a nevydá časové razítko.*

Pre vydávanie časových razítok sa používa protokol TSP (Time-Stamp Protocol), ktorý je špecifikovaný v RFC-3161 [6]. Pozostáva zo žiadosti o časové razítko a odpovedi na túto žiadosť. Po vložení protokolu do niektorého z transportných protokolov vznikne protokol typu klient-server. Je možné použiť protokol TCP alebo teoreticky i UDP na známych portoch pre TSP - 318. Najbežnejším spôsobom je však komunikácia pomocou protokolu HTTP alebo elektronickej pošty. V každom prípade je nutné definovať MIME hlavičku vo formáte:

```
Content-Type: application/timestamp-query
```

MIME hlavička odpovedi TSA musí obsahovať zase nasledujúci typ:

```
Content-Type: application/timestamp-reply
```

Časové razítka je možné vydávať aj v režime off-line, kedy klient zašle žiadosť TSA a tá vydá časové razítko s určitým časovým odstupom. K dispozíci väčšinou poskytuje každá TSA software na generovanie časového razítka, ktoré sa potom ukladá do samostatných súborov s koncovkou .tsr. V prípade ukladania žiadosti o časové razítko do samostatného súboru sa používa koncovka .tsq.

3.3 Žiadosť o časové razítko

Pri popise objektov v rámci oblasti PKI sa používa prevažne jazyk ANS.1 (Abstract Syntax Notation One), ktorý je ľahko čitateľný pre človeka. Pri komunikácii dvoch počítačov je naopak nutné zjednotiť tento tvar pre platformy procesorov, preto sa informácie prevádzajú z ANS.1 do kódovania BER (Basic Encoding Rules). Žiadosť o časové razítko je jednoduchá sekvencia, ktorá obsahuje nasledujúce položky popísane práve v jazyku ANS.1 podľa [2]:

```
TimeStampReq :: = SEQUENCE {
    version          INTEGER { v1(1) },
    messageImprint   MessageImprint,
    reqPolicy        TSAPolicyId                  OPTIONAL,
    nonce            INTEGER                      OPTIONAL,
    certReq          BOOLEAN                      DEFAULT FALSE,
    extensions       [0] IMPLICIT Extenstions     OPTIONAL
}
```

Nasleduje popis jednotlivých položiek:

`version` - obsahuje verziu protokolu TSP.
`messageImprint` - obsahuje objekt MessageImprint skladajúci sa z hashu dokumentu a z identifikátoru objektu algoritmu hashu:

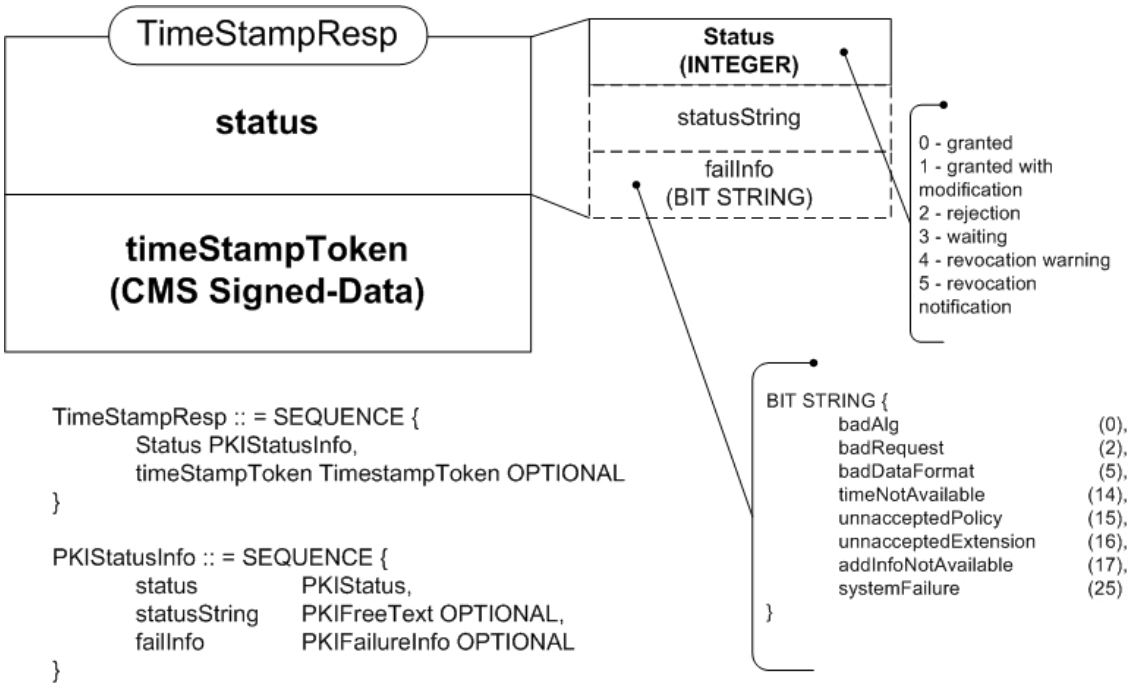
```
MessageImprint :: = SEQUENCE {
     hashAlgorithm  AlgorithmIdentifier,
     hashedMessage  OCTET STRING,
}
```

`reqPolicy` - voliteľná položka, ktorá môže obsahovať identifikátor objektu politiky, pod ktorou si žiadateľ praje vydať časové razítko.
`nonce` - voliteľná položka, ktorá obsahuje dostatočne veľké náhodné číslo k párovaniu žiadosti s odpoveďou. TSA túto hodnotu kopíruje aj do odpovede.
`certReq` - ak je táto voliteľná položka nastavená na hodnotu TRUE, značí to, že si žiadateľ praje aby odpoveď od TSA obsahovala taktiež certifikát TSA pre overenie pravosti časového razítka.
`extensions` - je obdobou rozšírenia certifikátu.

3.4 Odpoveď TSA

Odpoveď na žiadosť časového razítka od TSA je sekvencia obsahujúca `status` odpovede a prípadne vydané časové razítko. Tento `status` obsahuje číslo, ktoré popisuje výsledok operácie úspešnosti časového razítka. Môže nadobudnúť hodnoty nula, ktorá indikuje, že operácia orazítkovania dát je úspešná a odpoveď obsahuje časové razítko. Pokiaľ tento `status` obsahuje nenulovú hodnotu, došlo ku chybe a odpoveď TSA neobsahuje ani časové razítko. V tomto prípade

je v položke `failInfo` uvedený dôvod zamietnutia žiadosti a taktiež položka `statusString` obsahuje dôvody vo formáte UTF-8.



Obr.3.3: Štruktúra odpovedi TSA.

Samotné časové razítko je vložené do CMS správy `SignedData`. Táto pozostáva z položiek `contentType`, ktorá obsahuje identifikátor objektu špecifikujúci typ obsahu a `content`, ktorá zahŕňa samotnú správu. V prípade časového razítka obsahuje položka `contentType` správy CMS hodnotu `id-signedData`. Najdôležitejšími časťami sekcie `content` sú sekvencia `TSTInfo` a `SignerInfo`. `SignerInfo` obsahuje samotný podpis časového razítka a `TSTInfo` je sekvencia obsahujúca jadro časového razítka.

Štruktúra `TSTInfo` obsahuje nasledujúce položky:

```
TSTInfo :: = SEQUENCE {
    version          INTEGER { v1(1) },
    policy           TSAPolicyId,
    messageImprint   MessageImprint,
    serialNumber     INTEGER,
    genTime          GeneralizedTime,
    accuracy         Accuracy                    OPTIONAL,
    ordering         BOOLEAN                     DEFAULT FALSE,
    nonce            INTEGER                     OPTIONAL,
    tsa              [0] GeneralName             OPTIONAL,
    extensions       [1] IMPLICIT Extensions     OPTIONAL,
}
```

Popis jednotlivých položiek štruktúry `TSTInfo`:

`version` - obsahuje verziu protokolu TSP.
`policy` - obsahuje identifikátor objektu politiky TSA, pod ktorou bolo časové razítko vydané.
`messageImprint` - obsahuje hash, ktorý bol orazítkovaný, táto položka je kopírovaná zo žiadosti o časové razítko.
`serialNumber` - obsahuje poradové číslo vydaného časového razítka.
`genTime` - obsahuje čas vydania razítka. Je použitý UTC (Coordinated Universal Time) čas.
`accuracy` - táto položka vyjadruje presnosť času.

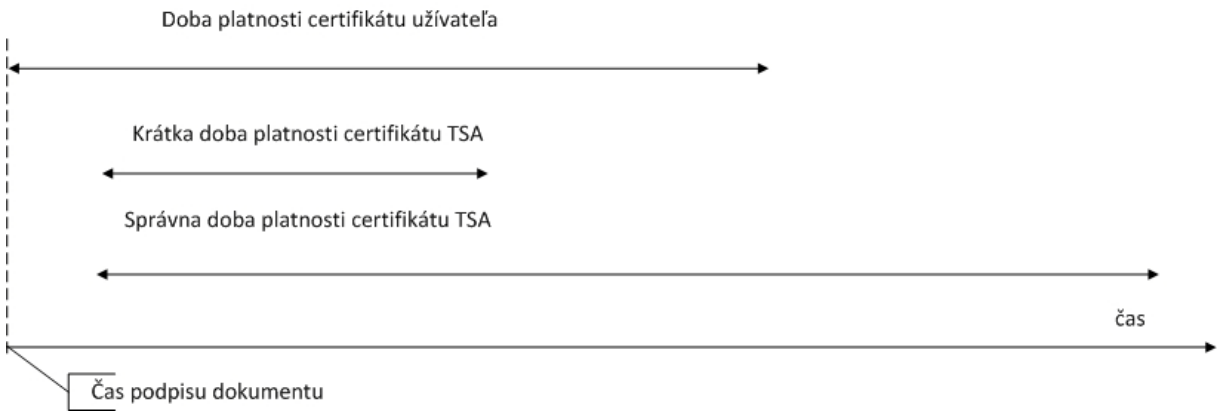
`ordering` - táto položka uvádza, či je časový údaj v časových razítkach tak presný, že by podľa neho mohli byť časové razítka zoradené v poradí v akom boli vydávané.
`nonce` - táto položka sa vyplní obsahom rovnomennej položky zo žiadosti o časové razítko a slúži k párovaniu odpovede a žiadosti o časové razítko.
`tsa` - položka môže obsahovať meno TSA.
`extensions` - položka je určená pre dodatočné informácie, ktoré sa v budúcnosti môžu pridávať do časového razítka.

3.5 Platnosť časového razítka

TSA je špeciálnym prípadom koncového uzla z hľadiska certifikačnej autority. Tá vydáva certifikát pre TSA s viacerými modifikáciami v porovnaní s bežnými koncovými užívateľmi. Najpodstatnejšou odlišnosťou v tomto prípade je odlišná doba platnosti certifikátu TSA v porovnaní s certifikátmi bežnými koncových užívateľov a nemožnosť použiť tento certifikát pre účely iné ako tvorbu časových razítiek.

Certifikát TSA máva spravidla rovnakú dobu platnosti ako certifikát CA, ktorá ho vydala. Časové razítko je možné overiť po dobu platnosti certifikátu samotnej TSA [2].

V prípade overovania digitálneho podpisu dokumentu, pre ktorý sme zaobstarali časové razíko musíme overiť časové razítko pomocou príslušného certifikátu TSA a následne overiť taktiež digitálny podpis pomocou príslušného certifikátu koncového užívateľa, ktorý daný podpis vytvoril.



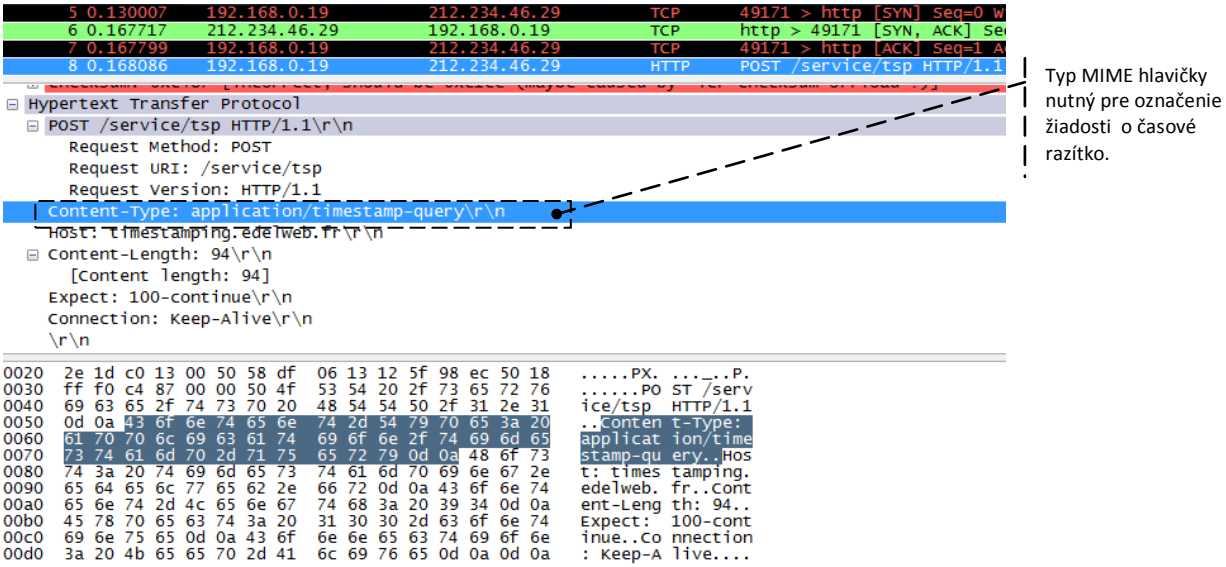
Obr.3.3: Doba platnosti časového razítka.

Pokiaľ by sa ale doba platnosti certifikátu TSA prekrývala s dobou platnosti certifikátu koncového užívateľa len o niekoľko dní, musel by sa obnoviť digitálny podpis dokumentu po týchto pár dňoch. Riešenie v tomto prípade teda spočíva v stanovení doby platnosti certifikátu TSA oveľa dlhšej ako je doba platnosti certifikátov bežných užívateľov (viz obr.3.3).

3.5 Analýza odpovede a žiadosti o časové razítko v praxi

Nasledujúce ukážky komunikácie typu klient-server pri dotazovaní sa o časové razítko na serveri TSA boli odchytené v programe wireshark. Jednotlivé polia objektov prenášaných správ sú prehľadne vyobrazené na obrázkoch 3.4-3.7, kde sú taktiež popísané najdôležitejšie sekvencie.

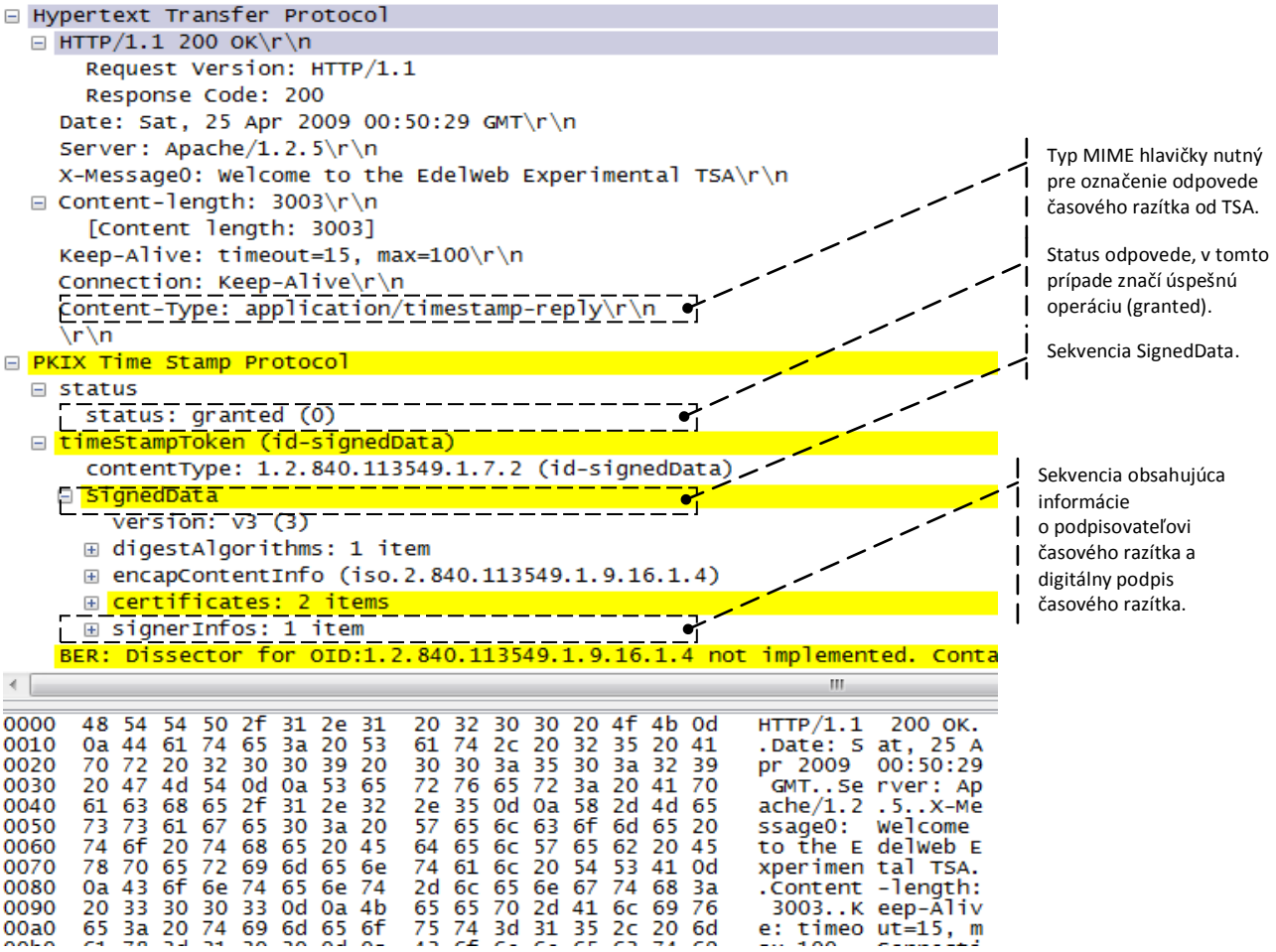
Žiadosť o časové razítko so správnym formátom MIME hlavičky protokolu HTTP:



Typ MIME hlavičky nutný pre označenie žiadosti o časové razítko.

Obr.3.4: Zachytený obsah paketu žiadosti o časové razítko.

Odpoveď TSA s viditeľnou základou štruktúrou objektu `satus` a `timeStampToken`:



Typ MIME hlavičky nutný pre označenie odpovede časového razítka od TSA.

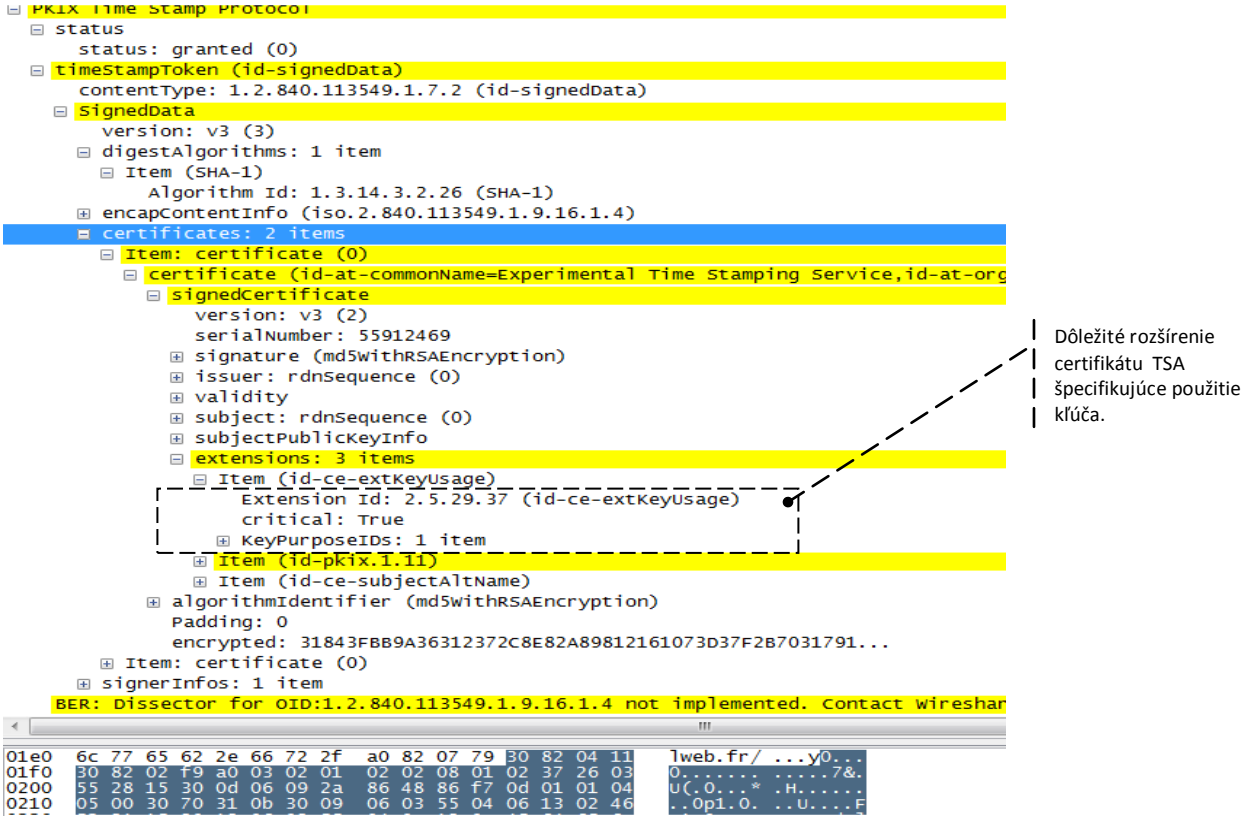
Status odpovede, v tomto prípade značí úspešnú operáciu (granted).

Sekvencia SignedData.

Sekvencia obsahujúca informácie o podpisovateľovi časového razítka a digitálny podpis časového razítka.

Obr.3.5: Zachytený obsah paketu odpovedi od TSA I.

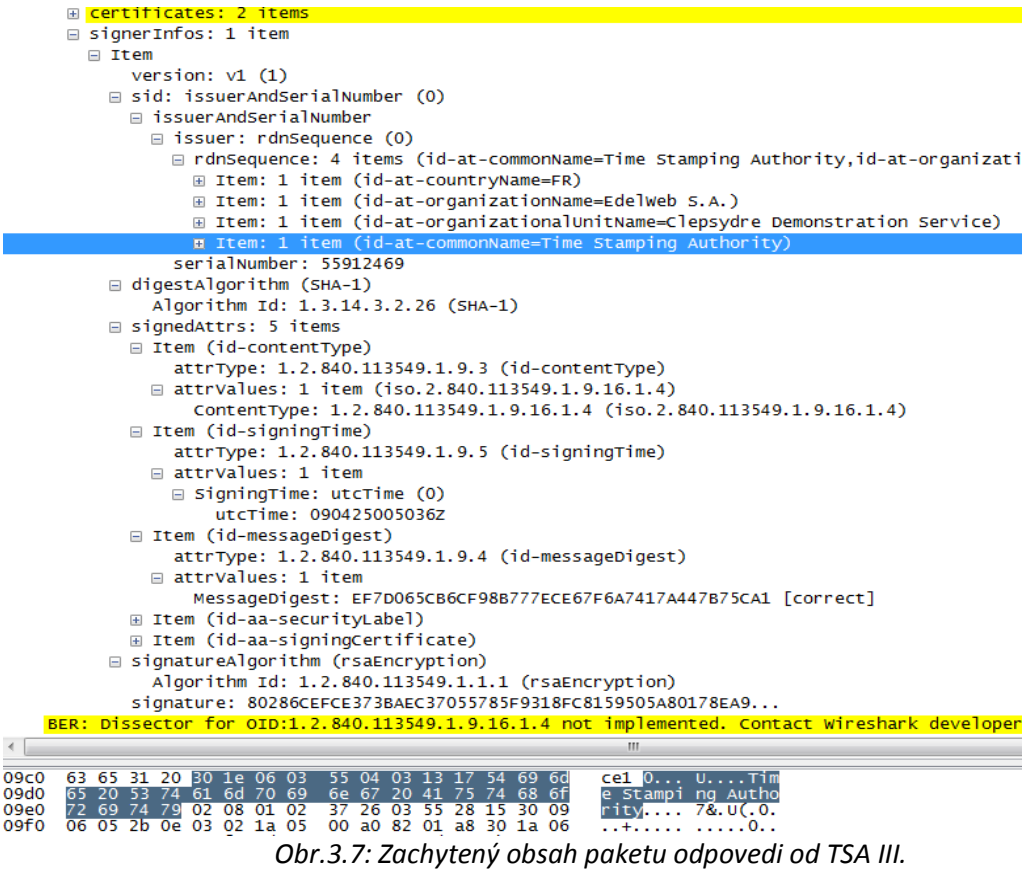
Odpoveď TSA s viditeľnou štruktúrou obsahu certifikátu TSA:



Dôležité rozšírenie certifikátu TSA špecifikujúce použitie kľúča.

Obr.3.6: Zachytený obsah paketu odpovedi od TSA II.

Odpoveď TSA s viditeľnou štruktúrou obsahu objektu `SignerInfo`:



Obr.3.7: Zachytený obsah paketu odpovedi od TSA III.

4 Infraštruktúra PKI

PKI (Public Key Infrastructure - infraštruktúra verejných kľúčov) je sústava technických a organizačných opatrení spojených s vydávaním, správou, používaním a revokovaním certifikátov verejných kľúčov.

Výraz PKI sa používa na opis procesov, technológií a praktík, ktoré by pri ich správnej implementácii a použití mali poskytnúť bezpečnú infraštruktúru. Táto infraštruktúra býva zabezpečená digitálnymi certifikátmi, kľúčmi a asymetrickou kryptografiou.

PKI poskytuje:

Autentizáciu – je možné ju definovať ako overenie identity objektu a je v PKI zabezpečená pomocou digitálnych certifikátov.

Dôvernosť – popisuje bezpečný prenos informácií cez nezabezpečené médium (internet) s istotou, že pri prenose sa k informáciám nedostala žiadna neoprávnená osoba.

Integritu – zaisťuje aby dáta pri prenose cez nezabezpečené médium neboli žiadnym spôsobom zmenené alebo modifikované. Integrita dát je v PKI zabezpečená pomocou jednosmerných funkcií - hash.

Nepopierateľnosť - zabezpečuje dôveryhodnosť pri zaistení vlastníctva nad digitálnym dokumentom a je v PKI zabezpečená pomocou digitálnych podpisov.

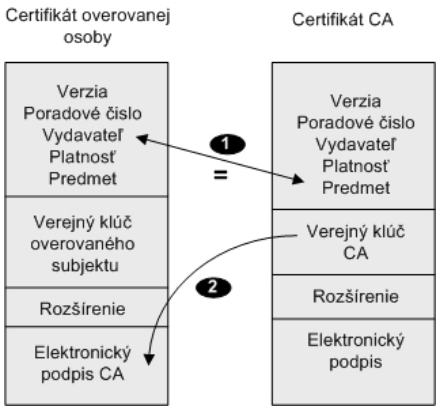
Kontrolu prístupu – zaoberá sa myšlienkou, aby existoval bezpečný prístup k dôveryhodným informáciám pre osoby, ktoré na to majú oprávnenia. PKI zabezpečuje túto kontrolu prístupu pomocou privátnych a verejných kľúčov.

4.1 Certifikačné a registračné autority

Komunikácia dvoch zúčastnených strán pri použití asymetrickej kryptografie ešte nezaručuje, že daná komunikácia nemôže byť žiadnym spôsobom narušená. Pokiaľ útočník podvrhne jednému z komunikujúcich svoj verejný kľuč, celý systém zabezpečenia pomocou asymetrickej kryptografie stráca význam. Práve proti podvrhnutiu verejného kľúča je vhodné sa brániť certifikáciou verejného kľúča nezávislou treťou stranou – certifikačnou autoritou [4].

Pokiaľ jeden z komunikujúcich má snahu overiť certifikát toho druhého s využitím certifikačnej autority, môže tak učiniť pri dodržaní nasledujúcich krokov:

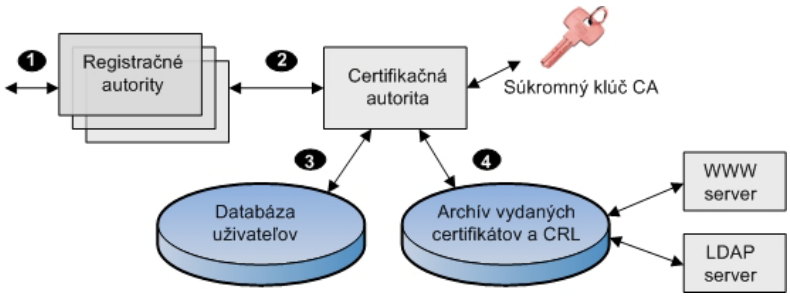
1. Overovaný certifikát má položku *Vydavateľ*. Overujúci si nájde vo svojom úložišti certifikátov dôveryhodných certifikačných autorít alebo iným spôsobom získa certifikát, ktorý má položku *Predmet* zhodnú z položkou *Vydavateľ* overovaného certifikátu.
2. Overujúci účastník komunikácie vyextrahuje verejný kľuč z certifikátu certifikačnej autority a pomocou neho overí elektronický podpis na certifikáte overovanej osoby.



Obr.4.1: Overenie certifikátu pomocou CA.

Na obr. 4.1 je znázornená väzba medzi certifikátom CA a overovanej osoby.

Certifikačná autorita je teda nezávislá tretia strana, ktorá vydáva certifikáty. Môžeme ju chápať buď ako aplikáciu vydávajúcu certifikáty alebo ako inštitúciu, ktorá zaisťuje proces vydávania certifikátov.

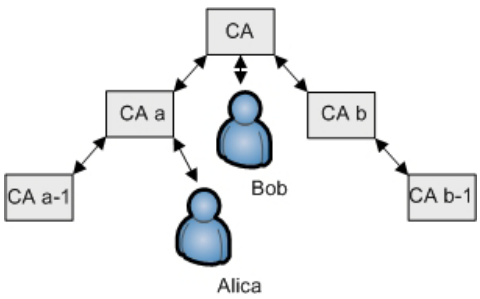


Obr.4.2: Štruktúra certifikačnej autority.

Inštitúcia certifikačnej autority sa skladá z nasledujúcich častí (viz obr.4.2):

1. Registračných autorít (RA) - overujú totožnosť žiadateľov a následne spostredkuvajú vydanie certifikátov. Môžu to taktiež byť servery, s ktorými žiadateľ výhradne komunikuje elektronicky.
2. CA ako aplikácia vydávajúca certifikáty podpísané súkromným kľúčom CA. Súkromný kľuč je v tomto prípade najväčším aktívom CA a tá si ho musí zodpovedne chrániť.
3. Databáza užívateľov.
4. Archív vydaných certifikátov a CRL prístupný cez webové rozhranie alebo protokol LDAP.

Certifikačná autorita môže byť iba koreňom v strome certifikačných autorít. Táto môže okrem klasických certifikátov pre koncových užívateľov vydávať aj certifikáty pre ďalšie podradené certifikačné autority a tieto podradené certifikačné autority zase môžu vydávať certifikáty pre ich podradené certifikačné autority. Výsledkom je teda stromová štruktúra certifikačných autorít (viz obr. 4.3).



Obr.4.3: Stromová štruktúra certifikačných autorít.

Certifikát podriadenej certifikačnej autority má na rozdiel od koreňového certifikátu rôzne hodnoty v položkách *Vydavateľ* a *Predmet*. Tieto certifikáty sa označujú ako krížové certifikáty podľa normy X.509. Tento je podobný klasickému užívateľskému certifikátu, avšak líši sa najmä v rozšíreniach certifikátu. Držiteľa krížového certifikátu je teda možné obmedziť pomocou špecifických príznakov napríklad vo vydávaní ďalších krížových certifikátov alebo iba na vydávanie certifikátov subjektom majúcim mená konkrétnych domén.

V prípade, že je potreba overiť certifikát koncového užívateľa, ktorý sa nachádza v zložitejšom hierarchickom usporiadaní certifikačných autorít je nutné preveriť nielen certifikát koncového užívateľa ale i certifikát certifikačnej autority, ktorá tento vydala. Pokiaľ táto certifikačná autorita má krížový certifikát je nutné overiť aj ten, z toho vyplýva, že neoverujeme jeden certifikát, ale celý reťazec certifikátov [2].

4.1.1 Dôvera medzi certifikačnými autoritami

Môže nastať situácia, kedy dvaja účastníci komunikácie majú certifikáty vydané od rôznych certifikačných autorít, ktoré však nie sú súčasťou jedného stromu certifikačných autorít. V tomto prípade sa dostáva do popredia otázka dôveryhodnosti medzi rôznymi CA.

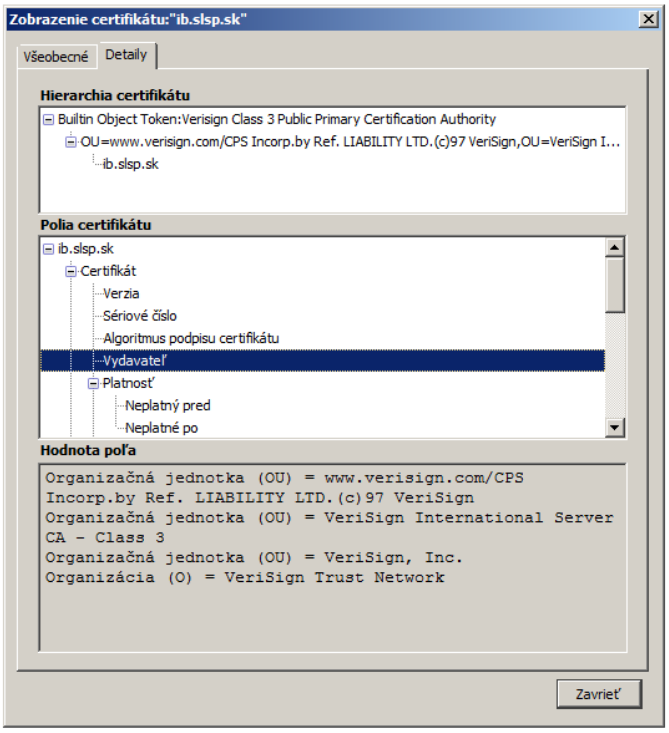
Tento problém má však niekoľko možností riešenia:

1. *Vzájomná krížová certifikácia medzi CA* - certifikačné autority, ktoré nepatria do toho istého stromu certifikačných autorít si vzájomne podpíšu svoje certifikáty.
2. *Koreňová CA tvoriaca most medzi stromami certifikačných autorít* - certifikačné autority, ktoré si chcú vzájomne dôverovať vybudujú koreňovú certifikačnú autoritu - most.
3. *CTL (Certificate Trusted List) - Zoznam dôveryhodných certifikátov* - vytvorenie a distribuovanie zoznamu certifikátov dôveryhodných koreňových certifikačných autorít podpísaný dôveryhodným objektom.

4.2 Certifikát

Je elektronický dokument, ktorý slúži k identifikácii jednotlivca, servera, alebo spoločnosti. Organizácia vydávajúca a zaoberajúca sa certifikátmi sa nazýva certifikačná autorita (CA). Je to organizácia potvrdzujúca pravosť, identitu spomínaných entít. Môže to byť aj fyzická osoba, právnická osoba alebo nejaká organizačná zložka štátu, ktorá vydáva certifikáty a vedie ich evidenciu, prípadne poskytuje ďalšie služby spojené s elektronickým podpisom. Certifikát obsahuje verejný kľúč vlastníka, názov vlastníka, dobu platnosti, názov CA, ktorý certifikát vydala, sériové číslo, prípadne ďalšie informácie. Najdôležitejšou informáciou certifikátu je digitály podpis CA, ktorá certifikát vydala. Autentizácia pomocou certifikátu je obecne spoľahlivejšia ako autentizácia pomocou hesla, pretože je založená na tom, že užívateľ niečo vlastní (súkromný kľúč) a nie na tom čo pozná (heslo) [4].

V súčasnej dobe existuje niekoľko noriem definujúcich štruktúru certifikátu. Pre potreby internetu sa vychádza zo štandardu X.509 verzie 3 a aktuálnym internetovým profilom certifikátu je dnes štandard RFC-3280 [7].



Obr.4.4: Obsah certifikátu.

Jednotlivé položky certifikátu (viz obr.4.4) možno popísať nasledovne:

Verzia certifikátu - definuje, či je certifikát normy X.509 odvodený z verzie 1, 2 alebo 3. V prípade verzie jedna má táto položka hodnotu nula, v prípade dva hodnotu jedna a v prípade tri hodnotu dva. V súčasnej dobe sa výhradne využívajú certifikáty verzie 3.

Sériové číslo - je to kladné celočíselné číslo, ktoré je jednoznačné v rámci konkrétnej certifikačnej autority, z toho vyplýva, že dvojica *Sériové číslo + Vydavateľ* jednoznačne určujú certifikát.

Algoritmus podpisu certifikátu - popisuje dvojicu algoritmov použitých certifikačnou autoritou pre vytvorenie digitálneho podpisu. Prvý z nich definuje algoritmus na tvorbu hashu a druhý spôsob šifrovania tohto hashu.

Platnosť od/do - tieto položky popisujú dobu platnosti certifikátu. Po skončení doby platnosti certifikátu, tento nie je ešte opovrhnutia hodný, pretože môže slúžiť k overeniu elektronického podpisu dokumentov v dobre platnosti certifikátu.

Vydavateľ - obsahuje jedinečné meno certifikačnej autority, ktorá certifikát vydala, preto je nutné aby samotná CA mala jedinečné meno v rámci všetkých CA.

Predmet - popisuje vlastníka certifikátu. Pri použití certifikátu normy X.509 verzie 3, musí byť predmet jedinečný v rámci všetkých objektov certifikovaných danou CA. Certifikačná autorita však môže vydať jednej a tej istej osobe certifikát s rovnakým predmetom, pretože sa jedná o tú istú osobu.

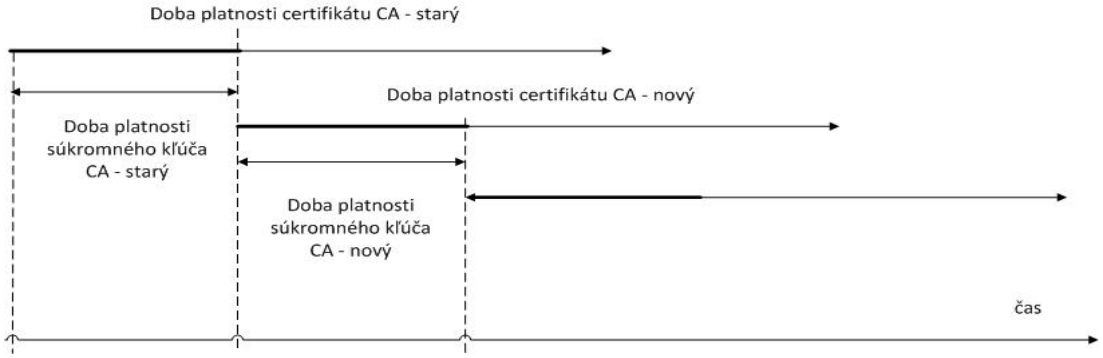
Pozn.: Obe položky Vydavateľ i Predmet majú rovnaký dátový formát označovaný ako jedinečné meno. Jedinečné meno je tvorené čiastkovými informáciami o objekte – relatívnymi jedinečnými menami. Relatívne jedinečné meno je množina atribútov tvorených dvojicou identifikátorom objektu (Country, Common Name atd.) a hodnotou (napr. CZ). Relatívne jedinečné

meno môže byť napr.: Common Name=Miroslav Remias. Jedinečné meno je potom sekvenciou relatívnych jedinečných mien: Common Name=Miroslav Remias, Organization=VUT, Country=CZ.

Verejný kľuč - skladá sa z dvoch informácii, identifikátoru algoritmu, pre ktorý je verejný kľuč určený a zo samotného verejného kľúča.

Rozšírenie certifikátu - obsahuje doplňujúce informácie certifikátu ako sú napríklad:

- *Platnosť súkromného kľúča* - umožňuje vyznačiť kratšiu dobu platnosti súkromného kľúča ako je doba platnosti certifikátu, čo umožňuje digitálne podpisovať dokumenty po kratšiu dobu ako overovať ich podpis (viz obr.4.5). Toto spravidla využívajú certifikačné autority.



Obr.4.5: Doba platnosti súkromného kľúča CA.

Po dobu platnosti súkromného kľúča certifikačnej autority je tento používaný k podpisovaniu vydávaných certifikátov. Po uplynutí tejto doby začne certifikačná autorita využívať nový certifikát, súkromný kľuč starého certifikátu CA je po uplynutí tejto doby zlikvidovaný, avšak užívatelia vlastniaci certifikát podpísaný týmto starým súkromným kľúčom majú možnosť si overiť naďalej svoj certifikát.

- *Použitie kľúča* - pomocou tohto rozšírenia je možné obmedziť spôsob využitia verejného kľúča certifikátu. Toto rozšírenie obsahuje reťazec bitov. Každý bit opovedá určitému spôsobu použitia certifikátu. Daný bit musí byť v tomto prípade nastavený na hodnotu TRUE, čo znamená, že certifikát je možné k danému účelu využívať. Ak sa však daný bit v reťazci nevyskytuje, predpokladá sa, že je nastavený na hodnotu TRUE.

  Popis jednotlivých bitov reťazca:

  - *Digital signature* - certifikát je určený k elektronickému podpisu dát. Takýto certifikát môže slúžiť k autentizácii užívateľov alebo k overeniu integrity dát.
  - *Non Repudiation* - certifikát je určený k overovaniu pravosti.
  - *Key Encipherment* - certifikát je určený k šifrovaniu kľúčov.
  - *Data Encipherment - verený kľuč tohto certifikátu je určený k šifrovaní dát.*
  - *Key Agreement - certifikát je určený pre výmenu kľúčov.*
  - *Key Certificate Sign - jedná sa o certifikát certifikačnej autority.*
  - *CRL Sign - verejný kľuč tohto certifikátu je určený k overovaniu CRL.*

- *Alternatívne meno predmetu* - umožňuje vložiť ďalšie jedinečné mená držiteľa certifikátu.
- *Certifikačné politiky* - toto rozšírenie obsahuje identifikátor dokumentu Certifikačná politika. Je to dokument, ktorý špecifikuje postupy, praktiky a ciele slúžiace k overeniu certifikátu pred tým, ako je použitý. Špecifikuje pravidlá, za ktorých sú vydávané certifikáty CA.

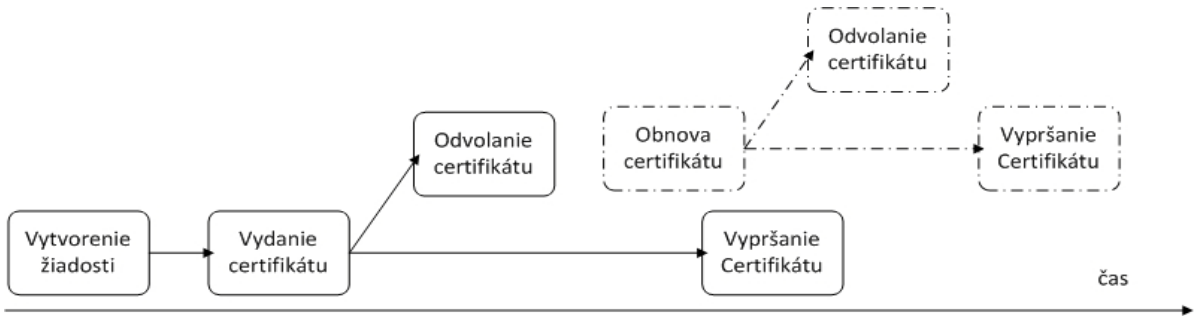
- *Identifikátor kľúča predmetu.*
- *Identifikátor kľúča úradu.*

Posledné dve položky rozšírenia certifikátu sa týkajú možnosti, kedy Certifikačná autorita má viac ako jeden certifikát. V prípade, kedy konkrétny z viacerých certifikátov CA bol použitý pre podpis certifikátu žiadateľa, sa musia tieto položky rovnať aby bolo možné špecifikovať, ktorý z certifikátov CA bol použitý.

4.2.1 Životný cyklus certifikátu

Tak ako väčšina obejkov vrámci PKI aj samotný certifikát prechádza v priebehu svojej existencie určitými fázami, ktoré tvoria jeho životný cyklus (viz obr.4.6). Ten pozostáva z nasledujúcich fáz:

1. *Vytvorenie žiadosti o certifikát* - toto môže ale nemusí predchádzať generovaniu párových dát, ktorých generovanie môže prevziať pri procese vydávania certifikátu samotná certifikačná autorita.
2. *Vydanie certifikátu* - prípadne môže byť certifikát verejne publikovaný.
3. *Platnosť certifikátu* - po tom ako bol certifikát vydaný, nie je nutnosťou jeho okamžitá platnosť. Tá je stanovená položkou *Not Before*, ktorá sa nachádza v certifikáte a končí vypršaním platnosti certifikátu alebo jeho predčasným odvolaním.
4. *Vypršanie platnosti certifikátu* - nastane po uplynutí doby, ktorá je stanovená v položke *Not After* uvedenej v certifikáte.
5. *Odvolanie certifikátu* - nastáva pred uplynutím doby jeho predpokladanej platnosti. Operáciu odvolania certifikátu prevádza certifikačná autorita tým, že zverejní certifikát na svojom zozname CRL. Myšlienka odvolania certifikátu môže pochádzať zo strany CA alebo na žiadosť samotného držiteľa certifikátu, či už to z dôvodu zničenia súkromného kľúča, jeho straty, odcudzenia alebo iných osobných dôvodov.



Obr.4.6: Životný cyklus certifikátu.

4.2.2 Žiadosť o certifikát

Žiadosť o certifikát by mala obsahovať nasledujúce položky:

- *Identifikačné údaje žiadateľa* – vo výslednom certifikáte budú uvedené ako položka predmetu certifikátu
- *Verejný kľuč vrátane identifikácie asymetrického algoritmu, pre ktorý je kľuč určený.*
- *Dôkaz o vlastníctve súkromného kľúča* – pokiaľ by certifikačná autorita vydala dva rovnaké certifikáty s verejným kľúčom dvom rôznym osobám, obe osoby by mohli jeden za druhého podpisovať dokumenty. Prípady, kedy si užívatelia vygenerujú rovnaký párový kľuč sú prakticky nemožné, v prípade použitia kvalitných generátorov náhodných čísiel.

CA kontroluje, či predložený verejný kľuč už v minulosti necertifikovala, ak áno, tak pôvodný certifikát odvolá a žiadosť zamietne.

- *Ďalšie údaje,* ktoré si užívateľ praje vložiť do certifikátu.
- *Heslá pre komunikáciu s certifikačnou autoritou:*
  - *jednorazové heslo pre vystavenie certifikátu* – užívateľ sa osobne dostaví do centra CA, preukáže svoju identitu, obdrží jednorazové heslo a potom ho použije pri žiadosti o certifikát cez internet.
  - *jednorazové heslo pre odvolanie certifikátu* – požíva sa pri odcudzení súkromného kľúča ako identifikácia užívateľa, ktorý chce zneplatniť svoj certifikát.
  - *trvalé heslo pre komunikáciu s CA.*

Nasledujúci postup platí všeobecne pri žiadosti o digitálny certifikát [1]:

- Pomocou dostupného softwarového vybavenia si žiadateľ vygeneruje párový kľuč.
- Žiadateľ si pripraví osobné identifikačné materiály nutné pre vydanie certifikátu (IČO, DIČ, resp. číslo OP, rodné číslo, adresu elektronické pošty apod.)
- Certifikačnej autorite predá žiadateľ dáta pre vydanie certifikátu spolu s dokladmi o ich pravosti (väčšinou v podobe správy zašifrovanej verejným kľúčom certifikačnej autority)
- Certifikačná autorita si overí pravosť údajov (žiadateľ sa dostaví do sídla certifikačnej, prípadne registračnej autority, kde predložením občianskeho preukazu potvrdí svoju totožnosť a pravosť zaslaných dát).
- Certifikačná autorita vytvorí digitálny dokument s príslušnými informáciami, ktorý následne podpíše svojim súkromným kľúčom a predá ho žiadateľovi.

4.2.3 Koreňový certifikát

Koreňový certifikát je taký, ktorý si vydáva sám žiadateľ o tento certifikát [2]. Má rovnakú štruktúru ako klasický certifikát X.509 verzie 1 alebo 3. Koreňový certifikát je možné identifikovať podľa toho, že položky *Predmet* a *Vydavateľ* certifikátu sú zhodné.

Ako dôkaz o vlastníctve súkromného kľúča užívateľom sa v koreňovom certifikáte použije digitálny podpis certifikátu, ktorý bol vytvorený súkromným kľúčom patriacim k verejnému kľúču uvedeného v certifikáte. Overenie teda prebieha verejným kľúčom, ktorý je uvedený priamo v tomto koreňovom certifikáte a toto overenie neprináša nič iné ako kontrolu integrity dátovej štruktúry certifikátu. Digitálny podpis koreňového certifikátu má z pohľadu jeho overenia nanajvýš hodnotu hashu a podvrhnúť koreňový certifikát je teda veľmi jednoduché. Tieto certifikáty musia byť teda distribuované dôveryhodnou cestou.

Koreňový certifikát v prípade CA je najvyššou autoritou, má síce rovnaký formát, avšak iný význam.

4.2.4 Kvalifikovaný certifikát

Kvalifikovaný certifikát je zvláštny typ certifikátu, ktorý má za účel nahradiť klasický (rukou písaný) podpis elektronickým podpisom, ktorý používa vo svojej legislatíve Európska únia [1].

Tento certifikát obsahuje identifikáciu držiteľa certifikátu založenom na oficiálnej identifikácii osoby, teda certifikačná autorita pozná konkrétnu osobu, ktorej certifikát vydala. Kvalifikovaný certifikát je teda zviazaný priamo a iba s fyzickou osobou.

Jednoznačná identifikácia je zaistená položkou predmetu certifikátu, dve rôzne osoby teda nemôžu mať túto položku zhodnú. Táto podmienka musí platiť po celú dobu existencie danej CA. Dosiahnutie tejto podmienky je možné aj s použitím hodnoty *serialNumber*, kedy v prípade dvoch rovnakých predmetov, sú dve osoby rozlíšené práve touto položkou.

Nutnou podmienkou existencie kvalifikovaných certifikátov sú taktiež jedinečné verejné kľúče, ktoré CA musí zaistiť pri vytváraní certifikátov. Pre certifikačnú autoritu to znamená archivovať verejné kľúče, ktoré žiadateľom podpísala aby ich mohla porovnávať, či už nie sú náhodou použité.

4.2.5 Zrušenie certifikátu

Certifikát vydaný certifikačnou autoritou má striktne danú dobu platnosti. Počas tejto prevádzkovej doby certifikátu môže dôjsť k jeho zrušeniu, či už z dôvodu zmeny údajov o subjekte, pre ktorý bol certifikát vydaný alebo z dôvodu straty, či odcudzenia súkromného kľúča držiteľa tohto certifikátu. Z tohto dôvodu má každá certifikačná autorita zoznam neplatných certifikátov (CRL – Certifikate Revocation List), ktorý sa aktualizuje spravidla v 6, 12, 18 alebo 24 hodinových intervaloch a je ho možné získať na stránkach danej certifikačnej autority. Prístup k CRL je dôležitý z dôvodu overovania platnosti certifikátu. Preto musí byť CRL pravidelne aktualizované [2].

Zoznamy zrušených certifikátov majú i svoje nevýhody. Mohla by nastať situácia, kedy v čase medzi dvoma aktualizáciami CRL skončí platnosť certifikátu. Aby sa tomuto predišlo bol navrhnutý nový spôsob zisťovania platnosti certifikátu pomocou protokolu OCSP (On-line Certificate Status Protocol).

Tento protokol umožňuje on-line prístup k informáciám o certifikátoch. OCSP prináša výhody pri overovaní platnosti certifikátu. OCSP je založené na databáze certifikátov, ktoré vracia v OCSP odpovedi aj hash overovacieho certifikátu. Hash má hlavný význam v odpovedi pre užívateľa, ktorý môže zistiť, že databáza obsahuje požadovaný certifikát a tým určiť bezpečne a jednoznačne aj stav platnosti certifikátu.

4.2.6 Certificate Revocation List

Ako už bolo spomenuté CRL môžeme chápať ako úradný list CA, na ktorom CA zverejňuje pravidelne odvolané certifikáty. Pokiaľ je už raz certifikát odvolaný mal by byť teda zverejňovaný aj na všetkých nasledujúcich CRL, pokiaľ nevyprší jeho pôvodná platnosť. CRL si spravidla vydávajú CA sami, avšak touto aktivitou môžu poveriť inú autoritu - autoritu pre vydávanie CRL. Tá potom vydáva takzvané nepriame CRL, kde je postup taký, že CA vydá certifikát serveru na vydávanie svojich CRL, ktorý obsahuje v rozšírení použitia kľúča bit *CRL Sign*. Na internete sa používa CRL vychádzajúca z normy X.509 verzie 2, ktorá je popísaná v dokumentácii RFC-3280 [7].

5 Podpora digitálneho podpisu a časového razítka v programovacom jazyku ASP.NET

ASP.NET (Active Server Pages .NET) je súčasťou .NET Framework technológie, ktorú vyvinul Microsoft v roku 2002 [5]. Táto technológia je postavená na princípe CLR (Common Language Runtime), teda podporuje vzájomnú kompatibilitu medzi jednotlivými programovacími jazykmi. CLR je virtuálny stroj, ktorý vykonáva všetok kód, ktorý je kompilovaný do univerzálneho jazyka - byte code (Microsoft Intermediate Language - MSIL), ktorý je podobný assembleru. Byte code je strojový kód virtuálneho stroja. Nezáleží teda v akom jazyku bude daná aplikácia naprogramovaná, všetky jazyky sú si plne rovnocenné. ASP.NET už nie je iba serverový skriptovací jazyk ako napríklad PHP, ale aplikačný programový jazyk, ktorý je možné použiť na strane servera na tvorbu kvalitných webových aplikácií. .NET Framework umožňuje napísať univerzálne potrebné funkcie v externom súbore vo zvolenom programovaciom jazyku a tieto súbory skompilovať do DLL knižníc, ktoré je potom možné ďalej využívať pre webovú aplikáciu, ale aj rôzne iné aplikácie. Tento jazyk má taktiež schopnosť vytvoriť a použiť znovu použiteľné komponenty, ktoré môžu obsahovať jednoduchú funkčnosť a tým redukovať množstvo kódu, ktoré musí programátor stránky napísať. ASP.NET stránky sú textové súbory s príponou .aspx a pozostávajú z kódu a značiek. Sú vytvorené dynamicky a spustené na serveri, ktorý renderuje odpoveď žiadajúcemu klientskému prehliadaču. V prípade žiadosti klienta o zdroje .aspx stránky, ASP.NET v priebehu programu rozoberie a vytvorí cieľový súbor do .NET Framework triedy. Táto trieda môže byť potom použitá, aby aktívne spracovala prichádzajúce žiadosti.

Podporované programovacie jazyky:

- ***Visual Basic .NET*** - jedná sa o syntax Visual Basic-u.
- ***C#*** - vyvinutý práve pre .NET Framework, jeho syntax je prispôsobená podľa dvoch najzastúpenejších a najpoužívanejších jazykov, C a Java.
- ***C++.NET*** – v súčasnej dobe je v praxi málo využívaný.
- ***J#*** - využívajú ho prevažne dlhoročný programátori Java.

5.1 Princíp funkčnosti ASP.NET

ASP.NET funguje podobne ako PHP, minimálne z pohľadu komunikácie medzi klientom a serverom. Požiadavka na zobrazenie web stránky je poslaná na server, ten vygeneruje čistý HTML kód, ktorý následne prepošle klientovi. Po príjme požiadavky od užívateľa webový server skontroluje požadovanú aplikáciu, či je spustená, a ak nie, naštartuje sa táto aplikácia a začne samostatne pracovať podľa požiadaviek užívateľa. Pokiaľ táto aplikácia dlhšiu dobu nie je požadovaná žiadnym užívateľom, automaticky sa vypne. Pri programovaní aplikácie sa dá naprogramovať, čo má aplikácia pri svojom spustení a jednotlivých krokoch spustenia vykonať a taktiež pri jej zániku. Programátor má teda plnú kontrolu nad každým krokom. Klasické serverové skriptovacie jazyky fungujú spôsobom, kedy je na servery spustená aplikácia s debbugerom pre tento jazyk a ten spracováva všetky požiadavky na tomto serveri, čím nastáva vyťažovanie servera a pri prípadnom páde tejto aplikácie sú znefunkčnené všetky webové aplikácie, zatiaľ čo pri ASP.NET len jedna aplikácia, teda len jeden web, ktorý sa ale pri ďalšom požiadavku reštartuje sám a pokračuje ďalej. Základný princíp vývoja ASP.NET webových aplikácii je založený na technológii formulárov a komunikáciu prostredníctvom nich.

Microsoft.NET poskytuje triedy, ktoré rozširujú kryptografické prostriedky poskytované Windows CryptoAPI rozhraním.

System.Security.Cryptography knižnica .NET Frameworku poskytuje triedy pre nasledujúce oblasti:

- Symetrické šifrovanie/dešifrovanie.
- Asymetrické šifrovanie/dešifrovanie.
- Hash.
- Digitálny podpis.
- Digitálny certifikát.
- XML podpis.

5.2 Hash

Podobne ako šifrovacie algoritmy sú hashovacie algoritmy implementované v .NET Frameworku s použitím troj-stupňovej dedičnosti. Hlavná trieda HashAlgorithm je abstraktná a obsahuje metódy pre preťaženie a výpočet hashu pre vloženú správu buďto vo formáte bytového poľa alebo streamu.

- Výpočet hodnoty hashu pre špecifikované bytové pole:
```
byte[] ComputeHash(byte[]);
```
- Výpočet hodnoty hashu pre špecifikovaný stream:
```
byte[] ComputeHash(Stream);
```
- Výpočet hodnoty hashu pre špecifikovanovú oblasť bytového pola:
```
byte[] ComputeHash(byte[], int, int);
```

Windows CryptoAPI podporuje iba MD5 a SHA1 hashovacie algoritmy a z tohto dôvodu sú implementované v .NET iba dve triedy na ich podporu - ***MD5CryptoServiceProvider*** a ***SHA1CryptoServiceProvider***. Všetky ostatné hashovacie funkcie sú implementované v riadenej triede, ktorá má štandardné riadené koncové pripojenie ako je napríklad ***SHA256Managed.***

.NET poskytuje nasledujúce hashovacie algoritmy:

- **MD5**CryptoServiceProvider (Message digest 5)
- **SHA1**CryptoServiceProvider (Secure hash algorithm s dĺžkou kľúča 160-bitov)
- **SHA256**Managed (Secure hash algorithm s dĺžkou kľúča 256-bitov)
- **SHA384**Managed (Secure hash algorithm s dĺžkou kľúča 384-bitov)
- **SHA512**Managed (Secure hash algorithm s dĺžkou kľúča 512-bitov)

5.3 Tvorba certifikátov X.509

5.3.1 Podpora digitálnych certifikátov v CryptoAPI

CryptoAPI má niekoľko preddefinovaných typov poskytovateľov úložíšť pre rôzne umiestnenia pričom fyzické umiestnenie je trvalé. Fyzický poskytovateľ úložišťa certifikátov poskytuje spojovanie certifikátov do skupín, zoznam odmietnutých (zrušených) certifikátov (Certificate Revocation List) a zoznam dôveryhodných certifikátov (Certificate Trust List).

Microsoft Windows poskytuje päť preddefinovaných logických úložíšť - MY, CA, TRUST, ROOT a USERS. Tieto úložištia sa nazývajú systémové a každý z nich môže obsahovať ďalšie logické úložištia certifikátov. Pokiaľ certifikát potrebuje byť členom niekoľkých rôznych logických skupín, zavádza sa pojem logické úložište certifikátov, ktoré predstavuje logickú skupinu fyzických úložíšť certifikátov. Všetky operácie, ktoré sú prevádzané nad logickým úložišťom sú prevedené na podradených fyzických úložištiach.

5.3.2 Podpora Cryptography v .Net Frameworku

Pre operácie s digitálnymi certifikátmi X.509v3 v .NET frameworku existuje podpora v podobe **System.Security.Cryptography.X509Certificates** knižnice s viacerými triedami. Táto podpora však nie je úplná a pre jednoduché uvedenie do problematiky poskytovaných možností nasledujú zhrnuté dôležité operácie, ktoré nie sú súčasťou preddefinovaných tried:

- Načítanie certifikátu z úložišťa certifikátov.
- Nájdenie súkromného kľúča patriaceho k verejnému kľúču digitálneho certifikátu z databáze kľúčov.
- Podpora pre čítanie X.509 v2 políčok certifikátu.
- Podpora pre čítanie X.509 v3 políčok certifikátu.
- Práca s CRL a CTL.

Pre prácu s digitálnymi certifikátmi poskytuje trieda ***X509Certificate*** iba statické metódy pre načítanie certifikátu zo súboru:

```
public static X509Certificate CreateFromCertFile(string filename);
```

5.3.3 Konzolové vybavenie pre tvorbu certifikátov

Pre samotnú tvorbu certifikátov Microsoft poskytuje iba konzolové aplikačné nástroje v .NET SDK, teda neexistuje programová podpora priamo v triedach .NET Frameworku:

Makecert	- Generuje X.509 certifikáty pre testovacie účely.
Certmgr	- Zhromažďuje certifikáty do CTL a môže byť taktiež použitý pre CRL.
Chktrust	- Overuje pravosť súboru podpísaného certifikátom X.509.

Nástroj pre tvorbu certifikátov (Makecert) generuje X.509 certifikáty pre testovacie účely a ukladá ich do certifikačného súboru vo zvolenom formáte. Generuje ako koreňový certifikát tak i certifikáty podpísané týmto koreňovým certifikátom.

Nasledujúci príklad prezentuje tvorbu X.509 DER certifikátu do súboru .cer v príkazovom riadku Microsoft Visual Studia 2003:

```
makecert -sk MirekR -n "CN=VUTBR" MirekR.cer
```

Pomocou tohto príkazu sa vytvorí certifikát (MirekR.cer) v osobnom adresári užívateľa, ktorý je momentálne prihlásený.

Makecert obsahuje základné a rozšírené príkazy. Rozšírené príkazy, ktoré podporujú vyššiu flexibilitu sa nachádzajú prehľadne vypísané v prílohe A. Základné príkazy sú zhrnuté v nasledujúcej tabuľke.

Tab.1: Základné príkazy nástroja Makecert.

Príkaz	Popis príkazu
-n x509name	Špecifikuje meno predmetu certifikátu. Je nutné špecifikovať meno v dvojitých uvodzovkách s predchádzajúcim výrazom CN=; napríklad: "CN=myName".
-pe	Označuje generovaný súkromný kľúč ako exportovateľný. Táto možnosť umožňuje aby bol súkromný kľúč obsiahnutý v certifikáte.
-sk keyname	Špecifikuje umiestnenie kľúča (key container) predmetu certifikátu, ktorý obsahuje súkromný kľúč. Pokiaľ umiestnenie kľúča neexistuje, bude vytvorené.
-sr location	Špecifikuje úložište certifikátu pre zadaný predmet certifikátu. Umiestnenie môže byť buďto **currentuser alebo localmachine.**
-ss store	Špecifikuje meno úložišťa pre zadaný predmet certifikátu, ktoré ukladá výstupný certifikát.
-# number	Špecifikuje sériové číslo od 1 do 2,147,483,647. Štandardná hodnota je unikátna.
-$ authority	Špecifikuje autoritu, ktorá podpisuje certifikát, ktorá musí byť nastavená na **commercial** alebo **individual** podľa druhu použitia.
-?	Zobrazí syntax príkazu a zoznam základných príkazov.
-!	Zobrazí syntax príkazu a zoznam rozšírených príkazov.

5.4 Podpis dát

Microsoft .NET podporuje nasledujúce dve triedy pre asymetrické šifrovanie/dešifrovanie:

- *RSA - System.Security.Cryptography.RSACryptoServiceProvider*
- *DSA - System.Security.Cryptography.DSACryptoServiceProvider*

RSA môže byť použité ako pre digitálny podpis dát tak pre šifrovanie/dešifrovanie. Na rozdiel od toho je DSA (Digital Signature Algorithm) striktne navrhnuté iba pre digitálny podpis a je pomalšie ako RSA. RSACryptoServiceProvider teda poskytuje dva sety metód, jednu pre šifrovanie súkromného kľúča a druhú pre podpis dát, na rozdiel od toho DSACryptoServiceProvider poskytuje metódy iba pre podpis dát. Rozdiel je možno taktiež nájsť v dĺžke podpisu. RSACryptoServiceProvider poskytuje 128 bajtový podpis, na rozdiel od 40 bajtového podpisu DSACryptoServiceProvideru. DSACryptoServiceProvider je rýchlejší pri výpočte podpisu a používa striktne iba SHA-1 hashovací algoritmus, kdežto RSACryptoServiceProvider je rýchlejší pri overovaní podpisu a používa viacero hashovacích algoritmov (SHA-1, MD5 atď.).

5.4.1 Digitalálny podpis pomocou RSACryptoServiceProvider

RSACryptoServiceProvider poskytuje dva sety metód pre digitálny podpis, jednu pre podpis hashu správy a druhú pre podpis čistého textu, z ktorého je najskorej vytvorený hash. Nasledujúci pár metód je použitý pre podpis hashu správy:

```
public byte[] SignHash(byte[] rgbHash, string str);
public bool VerifyHash(byte[] rgbHash, string str,byte[]
rgbSignature);
```

Pre podpis a overenie podpisu čistých dát je použitý nasledujúci pár metód:

```
public byte[] SignData(byte[] buffer,object halg);
public bool VerifyData(byte[] buffer,object halg,byte[] signature);
```

Tieto metódy majú dva páry metód na preťaženie, jedna z nich poberá ako vstup stream namiesto bajtového poľa a druhá poberá špecifický región dát bajtového poľa. Nasledujúci príklad ukazuje použitie metód v praxi:

```
string plaintext = "Správa na podpis";
byte[] plaintextbyte = new UnicodeEncoding().GetBytes(plaintext);
//Výpočet Hashu správy a následny podpis pomocou SignData metódy.
byte[] sign = _rsa.SignData(plaintextbyte,"MD5");
//Overenie podpisu pomocou VerifyData metódy.
bool test = _rsa.VerifyData(plaintextbyte,"MD5",sign);
```

5.4.2 Digitalálny podpis pomocou DSACryptoServiceProvider

DSACryptoServiceProvider obsahuje rovnaký set metód pre ako RSACryptoServiceProvider, na rozdiel však obsahuje navyše aj další pár metód pre digitálny podpis:

```
public override byte[] CreateSignature(byte[] rgbHash);
public override bool VerifySignature(byte[] rgbHash,byte[]
rgbSignature);
```

5.4.3 Digitalálny podpis pomocou tried Formatter a Deformatter

System.Security.Cryptography knižnica podporuje taktiež pomocné triedy pre digitálny podpis. Prípona Formatter je použitá v prípade tvorby digitálneho podpisu, na overenie podpisu je zase použitá prípona Deformatter.

Nasledjúci kód popisuje tvorbu podpisu pomocou triedy Formatter:

```
DSACryptoServiceProvider DSA = new DSACryptoServiceProvider();
DSASignatureFormatter dsfm = new DSASignatureFormatter(DSA);
dsfm.SetHashAlgorithm("SHA1");
byte[] SignedHash = dsfm.CreateSignature(hash);
```

Overenie podpisu je realizované pomocou triedy Deformatter:

```
DSACryptoServiceProvider DSA = new DSACryptoServiceProvider();
DSASignatureDeformatter dsdfm = new DSASignatureDeformatter(DSA);
dsdfm.SetHashAlgorithm("SHA1");
dsdfm.VerifySignature(hash, signature);
```

5.5 Časové razítko

Microsoft .NET Framework nemá implementovanú podporu pre časové razítka a ani podporu pre posielanie požiadavkou na server TSA. Z tohto dôvodu bude daná funkcionalita zabezpečená programovo v kóde. Bude teda vytvorená alebo pužitá trieda, ktorá bude produkovať požiadavky na orazítkovanie dát TSA vo formáte odporučenia podľa RFC3161 a taktiež bude musieť spracovávať odpovede na tieto požiadavky.

6 Webová aplikácia CAT

Webové aplikácie sa používajú na jednoduchých webových stránkach rôznych foto albumov, diskusných fór, ale aj v zložitejších prípadoch ako je internetové bankovníctvo. Z dôležitosťou týchto aplikácii rastie aj potreba ich ochrany pred neautorizovaným použitím, zaistením dátovej integrity a inými bezpečnostnými aspektmi. V reálnom svete sú tieto webové aplikácie navrhované pre používanie práve laickou verejnosťou, preto je veľmi dôležité klásť dôraz na ich jednoduchosť a prehľadnosť a bezpečnosť. Pre korektnú prácu s týmito, často krát rozľahlými systémami, je nutné dodržať správne zásady pre autentizáciu užívateľov tohto systému, zaistiť ich správu.

Webová aplikácia CAT (Certificate Authority and Timestamping) sa zameriava na aplikáciu doposiaľ nadobudnutých vedomostí z oblasti PKI a ASP.NET programovacieho jazyka. Aplikácia umožňuje registrovaným užívateľom, ktorý majú nárok si v systéme vytvoriť práve jeden kryptografický pár kľúčov, digitálne podpisovať dokumenty, správy a tieto odosielať vo forme e-mailových správ. Následne si ktokoľvek môže overiť pravosť digitálneho podpisu prijatej správy. Pri podpisovaní dokumentov, správ je tu možnosť vytvoriť k dokumentu časové razítko, ktoré jednoznačne zväzuje podpísaný dokument s časom jeho podpisu. Aplikácia taktiež dynamicky generuje kvázi CRL a CTL.

V nasledujúcich častiach tejto kapitoly sú stručne predstavené najpoužívanejšie funkcie a najdôležitejšie zdrojové kódy použité v tejto webovej aplikácii, ich vysvetlenie a celkový popis funkčnosti aplikácie.

6.1 Autentizácia užívateľov

Overovanie identity komunikujúcich účastníkov v aplikácii je kritický aspekt procesu prihlasovania sa do samotnej aplikácie. Je to overenie pravosti identity užívateľa, či je to skutočne tá osoba, za ktorú sa vydáva. Tak ako u väčšiny webových aplikácii aj táto je založená na znalosti užívateľského mena a hesla. Po úspešnom dokazovaní tajnej znalosti je užívateľovi pridelené takzvané autentizačné Cookies, ktoré sa ďalej používa počas práce v systéme. Pokiaľ by sa útočník dostal nedopatrením do systému a vydával by sa za niekoho, kým v skutočnosti nie je, mohol by digitálne podpisovať dokumenty v mene osoby, ktorá bola týmto zneužitá.

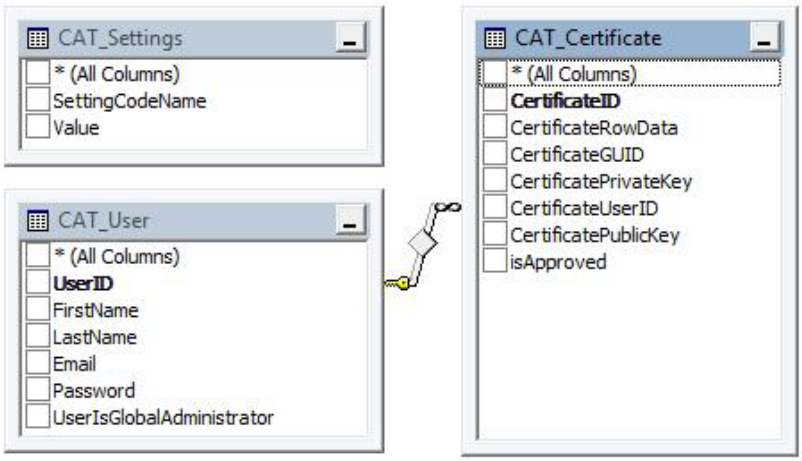
Obr.6.1: Úvodné prihlasovacie/registračné okno do aplikácie.

Prihlasovacia tajnosť - heslo, je zvyčajne ukladané do databáze. Heslo by sa nikdy nemalo vyskytnúť ako čitateľná informácia pre človeka, pretože pri neopatrnosti by útočník mohol preniknúť do databáze a zneužiť teda prihlasovacie údaje. Preto medzi preferované metódy ukladania hesla do databáze patrí použitie hashu hesla, ktorý jednoznačne popisuje toto heslo a pritom je v nečitateľnej podobe pre človeka. V prípade tejto aplikácie je požadované pri registrácii (viz obr.6.1) použiť 8-miestne heslo následne z ktorého je vytvorený hash a iba tento uložený ako tajná informácia korešpondujúca k danému užívateľovi. Keďže už samotná registrácia vyžaduje od užívateľa vložiť takto citlivé údaje, celá komunikácia sa od tohto bodu nachádza v bezpečnom režime SSL ako je možné vidieť na obrázku 6.1.

Z predchádzajúcich faktov je možné usúdiť, že implementácia správneho a bezpečného prihlasovania sa do aplikácie je dôležitá pre bezpečnosť práce užívateľov a predchádzanie napadnuteľnosti aplikácie.

6.2 Schéma databáze

Po väčšine sú všetky dnešné dynamické webové aplikácie, ktoré nejakým spôsobom uchovávajú dáta, postavené na použití databáz, či už to na platforme MS SQL alebo MySQL. Webová aplikácia CAT používa taktiež databázu MS SQL pre nevyhnutné uchovávanie dát o užívateľoch systému a ich certifikátoch. Databáza predstavuje tabuľky *CAT_User*, *CAT_Certificate* a *CAT_Settings*. Štruktúru jednotlivých tabuliek je možné prehľadne vidieť na obrázku 6.2.



Obr.6.2: Schéma databáze aplikácie CAT.

6.3 Správa užívateľov

V samotnom systéme môžu pracovať registrovaný, neregistrovaný užívatelia a administrátor(i) systému. Význam pre rozdelenie užívateľov do jednotlivých oblastí je možné pochopiť z nasledujúceho popisu:

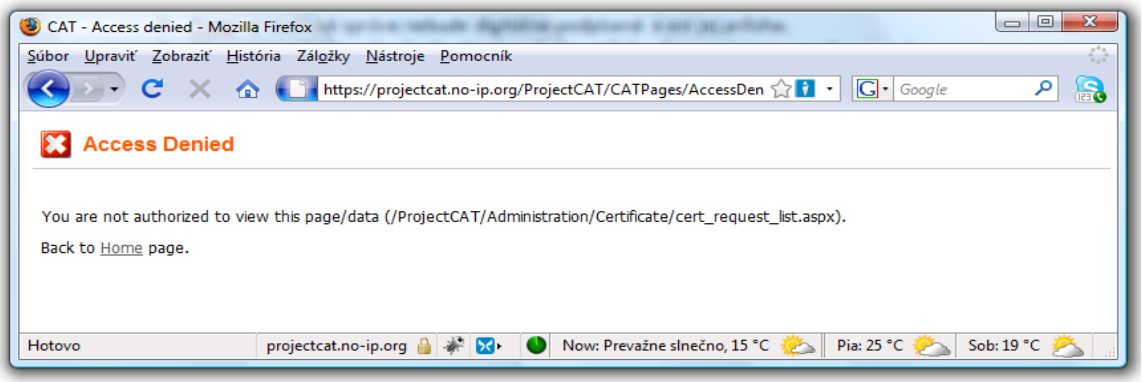
Neregistrovaný užívatelia – majú prístup k CTL a CRL CA, môžu overovať pravosť a platnosť digitálneho podpisu.

Registrovaný užívatelia – majú totožné vlastnosti s neregistrovanými užívateľmi a môžu navyše spravovať svoj profil, generovať žiadosť o certifikát, po schválení žiadosti administrátorom môžu digitálne podpisovať dokumenty s možnosťou časového razítka a následného poslania v e-mailovej správe.

Administrátor systému – navyše k registrovaným a neregistrovaným užívateľom môže spravovať užívateľské účty, vytvárať editovať a mazať užívateľov, zamietať alebo potvrdzovať žiadosti užívateľov o certifikáty, mazať certifikáty a spravovať celkové nastavenia systému.

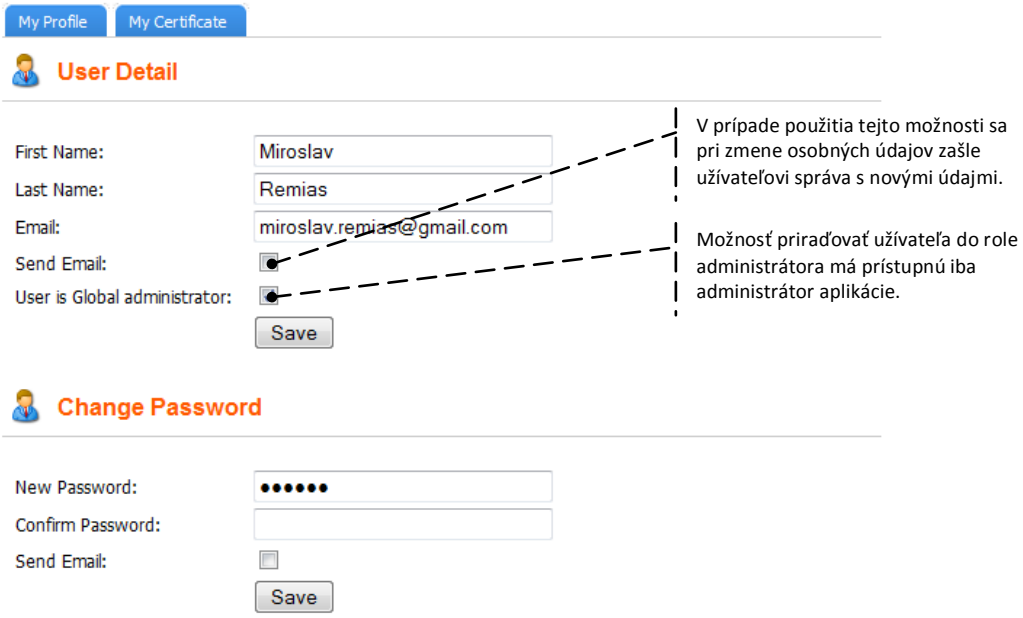
Každý užívateľ je viazaný práve na jeden kryptografický pár kľúčov, jeden certifikát v aplikácii. Primárnym kľúčom každého užívateľa či už v aplikácii alebo v databáze je jeho jedinečné identifikačné číslo - *UserID*. Tento identifikátor, ktorý sa nachádza v tabuľke *CAT_User* zväzuje teda užívateľa a certifikát k nemu prislúchajúci pomocou stĺpca *CertificateUserID* z tabuľky *CAT_Certificate*. Jedinečná je taktiež stanovená e-mailová adresa každého užívateľa podľa ktorej sa overuje digitálny podpis.

Práva na prístup užívateľov k jednotlivým modulom, stránkam alebo dátam aplikácie sú taktiež kontrolované na základe ich role. Inými slovami povedané, každý užívateľ má sprístupnené pracovať iba s takými možnosťami, ktoré boli popísané vyššie pri jeho roli. Pokiaľ by sa užívateľ snažil pristúpiť na základe znalosti danej URL k nepovoleným zdrojom bude okamžite oboznámený s týmto priestupkom (viz obr.6.3) a prístup mu bude zamedzený k tomuto zdroju.



Obr.6.3: Schéma databáze aplikácie CAT.

Tento prípad sa vyskytuje v aplikácii pokiaľ sa užívateľ snaží pristúpiť ku zdrojom prislúchajúcim práve administrátorovi systému.



Obr.6.4: Osobné údaje užívateľa.

V prípade, že užívateľ nie je prihlásený a pristupuje ku zdrojom prislúchajúcim registrovaným užívateľom, je najskôr presmerovaný na prihlasovaciu stránku (viz obr.6.1), kde je po úspešnom prihlásený opäť presmerovaný na stránku, ktorú vyžadoval.

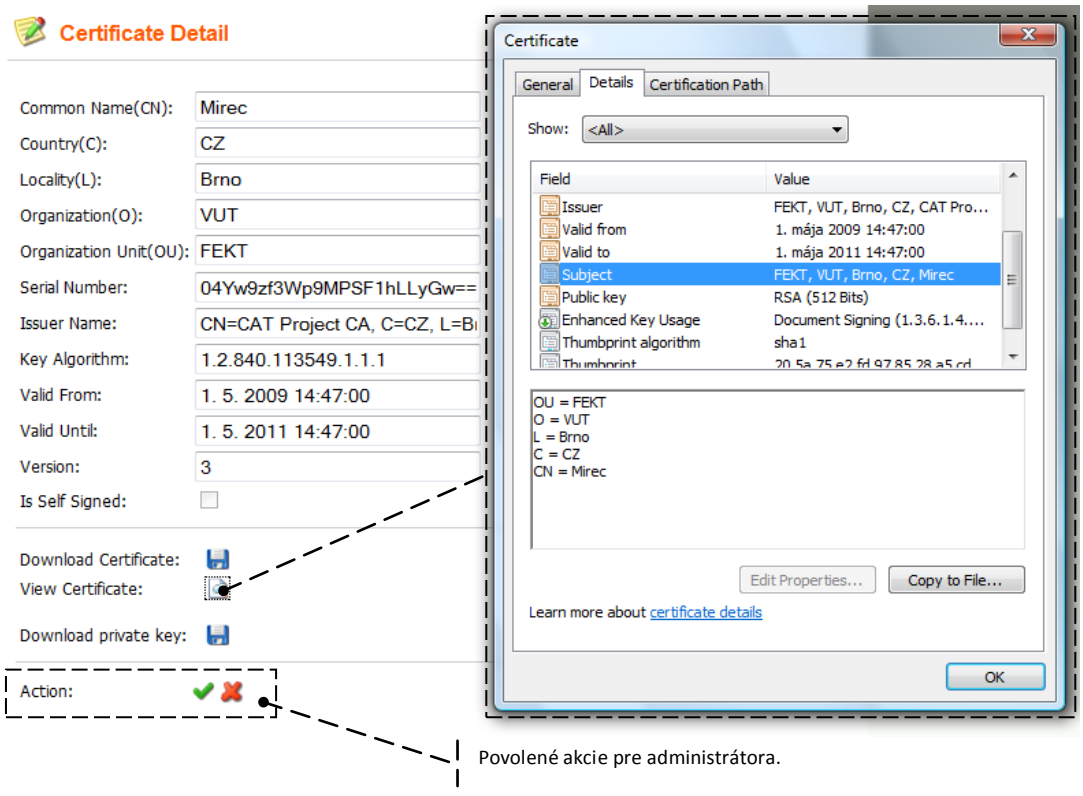
Registrovaný užívatelia majú taktiež možnosť meniť svoje osobné údaje v sekcii *My Profile* ako je možné vidieť na obrázku 6.4. Pri zmene údajov majú možnosť preposlať si tieto pozmenené osobné údaje alebo heslo do svojej e-mailovej schránky.

6.4 Generovanie certifikátu

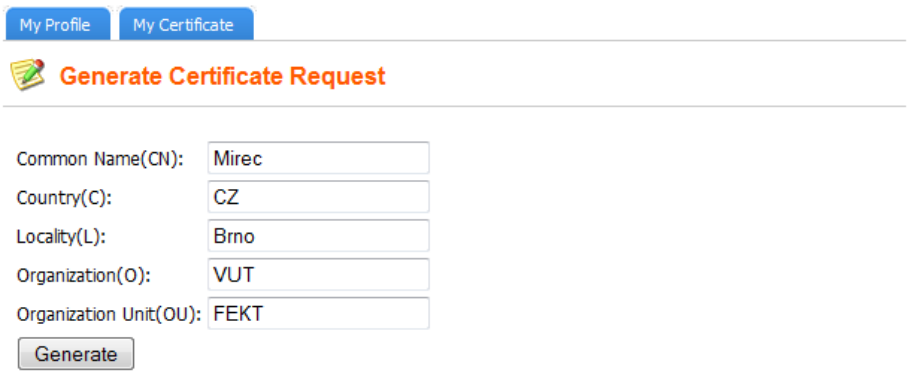
Pokiaľ si užívateľ ešte nevygeneroval vlastný certifikát, má tuto možnosť v sekcii *My Certificate*. K dispozícii má možnosť vyplniť predmet certifikátu (viz obr.6.6), ktorý v tomto prípade pozostáva z jedinečného mena, ktoré je sekvenciou relatívnych jedinečných mien ako sú *Common Name*, *Country*, *Locality*, *Organization* a *Organization Unit*. Pri generovaní certifikátu je užívateľovi vytvorený RSA kryptografický pár kľúčov o dĺžke 512 bitov a certifikát je automaticky podpísaný súkromným kľúčom CA. Zvolená dĺžka kľúča 512 bitov je z dôvodu jeho neskoršieho použitia pri práci s TSA a časovým razítkom, kde je táto podmienka nutná v spolupráci so zvolenou TSA. Takto vygenerované certifikáty užívateľov sú obmedzené na použitie iba pre digitálny podpis.

Informácia o certifikátoch je uložená v databáze v tabuľke *CAT_Certificate*. Samotné certifikáty sú uložené buďto v binárnej podobe v tejto tabuľke v stĺpci *CertificateRowData* alebo sa nachádzajú na disku servera v adresári *CATCertificate* pod jedinečným názvom vo formáte [GUID].cer, kde *GUID* (Globally Unique Identifier) je jedinečný 32 znakový identifikátor.

Administrátor systému následne vidí všetky žiadosti o certifikáty v sekcii *Certificate – list*, kde ich má možnosť zamietnuť, vymazať alebo potvrdiť. Užívateľ má taktiež možnosť nahliadnuť detailne k svojmu certifikátu, zobraziť si ho, stiahnuť a taktiež si stiahnuť svoj súkromný kľuč, ako je možné vidieť na obrázku 6.5.



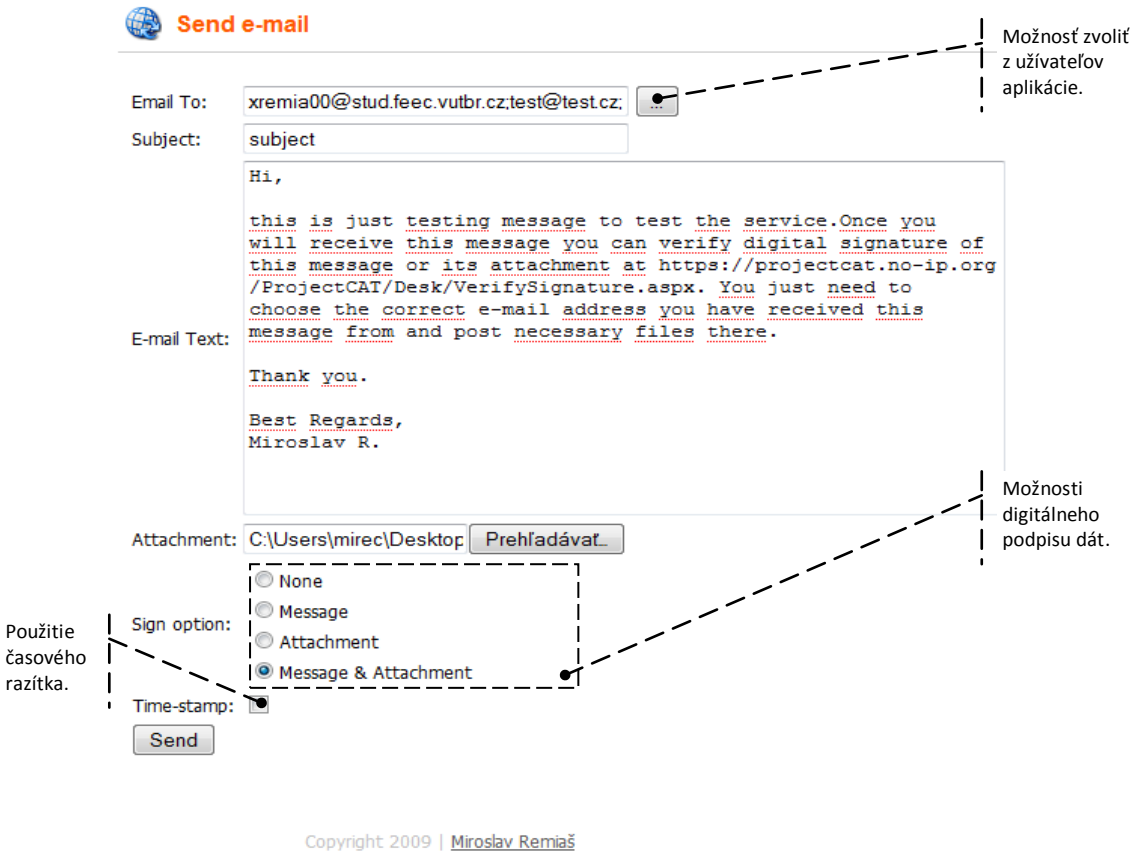
Obr.6.5: Detailný náhľad na vygenerovaný certifikát.



Obr.6.6: Generovanie certifikátu.

6.5 Digitálny podpis a časové razítko

V prípade, že bola užívateľovi kladne potvrdená žiadosť o certifikát administrátorom systému alebo je tento certifikát globálne platný, má užívateľ sprístupnený komunikačný modul (viz obr.6.7) pre posielanie e-mailových správ s možnosťou digitálneho podpisu a časového razítka. Užívateľ má možnosť zvoliť v tejto sekcii komu bude daná správa doručená v položke `Email To`. Má na výber z užívateľov aplikácie, ale môže samozrejme vložiť aj vlastnú e-mailovú adresu či použiť viacej e-mailových adries prijímateľov. Položky `Subject` a `E-mail Text` tvoria telo e-mailovej správy.

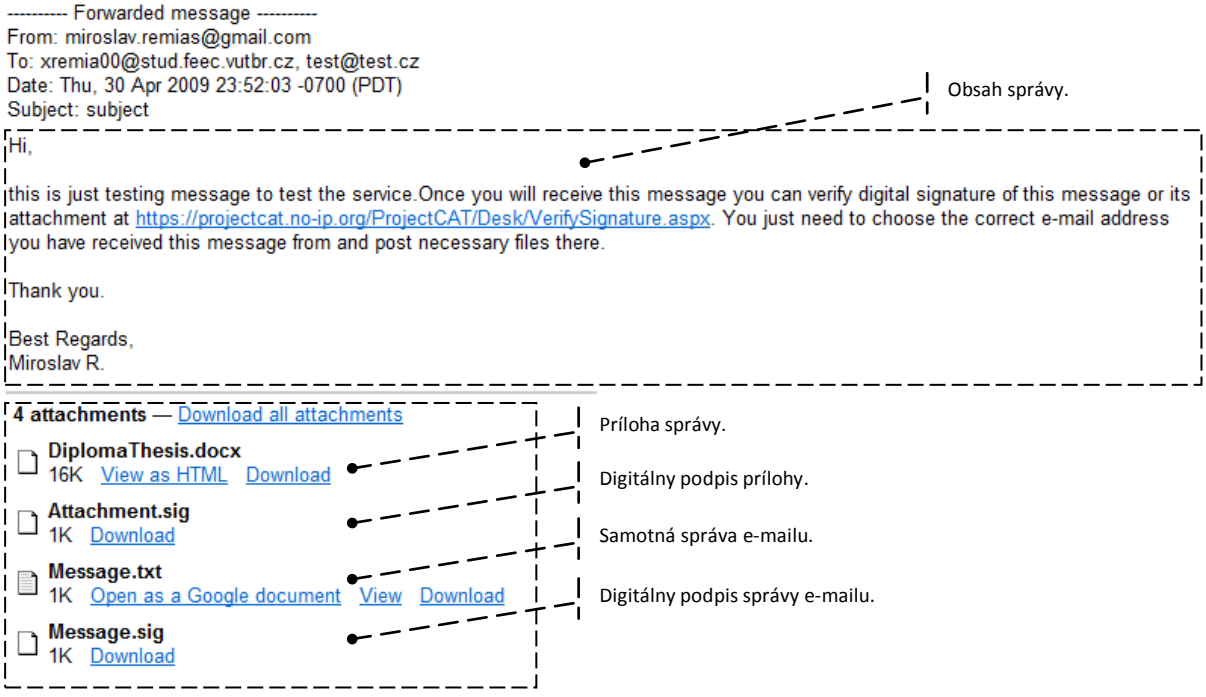


Obr.6.7: Digitálny podpis dokumentov spojený s e-mailovou komunikáciou.

Ďalej má užívateľ možnosť pripojiť jednu prílohu - súbor, dokument. Na výber má taktiež k dispozícií štyri možnosti digitálneho podpisu:

- *None* - e-mailová správa nebude digitálne podpísaná a ani jej príloha, teda sa jedná sa o klasickú e-mailovú správu.
- *Message* - prijatá správa bude obsahovať dve prílohy. Samotný text e-mailovej správy v súbore `Message.txt` a digitálny podpis tohto textu v podobe súboru s názvom `Message.sig`, ktorý prislúcha danému textu.
- *Attachment* - prijatá správa bude obsahovať dve prílohy. Samotnú prílohu vloženú užívateľom a digitálny podpis v podobe súboru s názvom `Attachment.sig`, ktorý prislúcha danej prílohe.
- *Message & Attachment* - je kombinácia predchádzajúcich dvoch možností, z čoho vyplýva, že prijatá správa bude obsahovať štyri prílohy - dva dátové súbory a dva súbory podpisov k nim prislúchajúce.

Ako je vidieť na obrázku 6.8, prijatá e-mailová správa obsahuje všetky potrebné informácie - dáta k jej neskoršiemu overeniu. Pri použití možnosti časového razítka sa digitálny podpis, či už správy e-mailu alebo prílohy, ešte pred tým ako je pripojený k e-mailovej správe, odošle autorite časových razítok, ktorá vráti samotné časové razítko podpisu dokumentu alebo správy. Prijatá e-mailová správa teda bude obsahovať rovnomenné prílohy ako v prípade použitia iba digitálneho podpisu, avšak tieto súbory budú zakončené koncouvkou .tsr, ktorá predstavuje práve odpoveď na žiadosť o časové razítko. Aplikácia používa takú autoritu časových razítok, ktorá je definovaná v nastaveniach aplikácie.



Obr.6.8: Príklad prijatia digitálne podpísanej e-mailovej správy spolu s podpisom prílohy.

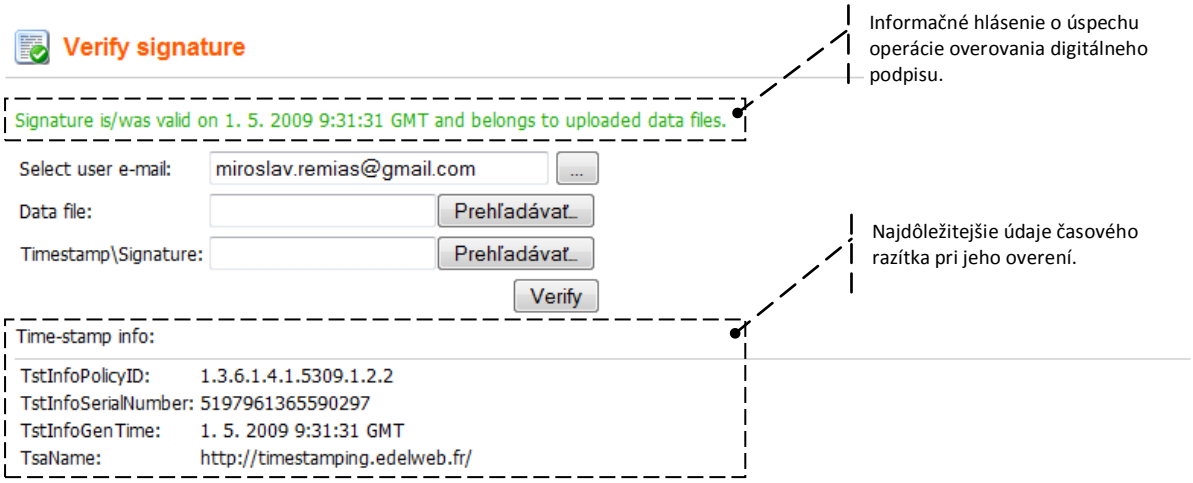
6.6 Overenie pravosti a platnosti digitálneho podpisu

Overenie, či daný podpis skutočne prislúcha dokumentu a užívateľovi, od ktorého prišla e-mailová správa s týmito dátami je možné uskutočniť v sekcii *Verify Signature* ako ukazuje obrázok 6.9. Zvolením správnej jedinečnej e-mailovej adresy, od ktorej nám prišla správa a vložením

patričných dátových súborov je teda možné overiť digitálny podpis. Aplikácia informuje o úspešnosti prevedenia tejto overovacej procedúry s patričnou výstupnou hlážkou. Overenie podpisu prebehne aj v prípade, že certifikát užívateľa už nie je platný, ale aj o tejto operácii je užívateľ informovaný a dôveruje teda podpisu na vlastné riziko.

Obr.6.9: Overenie pravosti digitálneho podpisu.

V tomto prípade je teda vhodnejšie použiť možnosť časového razítka. Časové razítko sa taktiež overuje v tejto sekcii na základe certifikátu autority časových razítok, ktorý je už obsahom časového razítka a ďalej rovnako ako v prípade digitálneho podpisu na základe jedinečnej e-mailovej adresy a s ňou spojeným certifikátom užívateľa. V prípade overenia podpisu pomocou vloženia časového razítka ponúka aplikácia dodatočné informácie o čase podpisu dokumentu, sériovom čísle razítka, mene TSA a jej politiky, ktoré je možno vidieť na obrázku 6.10.



Obr.6.10: Overenie pravosti digitálneho podpisu na základe časového razítka.

6.7 Nastavenia systému

Administrátor systému má sprístupnenú ponuku pre nastavenia aplikácie. V tejto sekcii je možné nastaviť nasledujúce hodnoty:

- *SMTP server name* - názov SMTP serveru vrátane portu, ktorý bude aplikácia využívať pre posielanie e-mailových správ.
- *SMTP user name* - uživateľské meno pre prístup na SMTP server.
- *SMTP user password* - uživateľské heslo pre prístup na SMTP server.
- *SMTP - use SSL* - možnosť zapnúť zabezpečený mód pri komunikácii s SMTP serverom.
- *Access Denied URL* - URL stránky, ktorá sa zobrazí pri prístupe do sekcie, ktorú daný užívateľ nemá právo navštíviť.
- *TSA URL* - URL poskytovateľa TSA na ktorej beží služba razítkovania posielaných dát.
- *TSA Login* - uživateľské meno pre prístup na TSA server ak je vyžadované.
- *TSA Password* - uživateľské heslo pre prístup na TSA server ak je vyžadované.
- *Store certificates in database* - ak je táto voľba použitá, všetky certifikáty užívateľov sú uložené v binárnej podobe v databáze.

Nastavenia aplikácie sú uchované v tabuľke *CAT_Settings* v databáze, kde stĺpec *SettingCodeName* obsahuje kódové meno nastavenia a stĺpec *Value* obsahuje samotnú hodnotu nastavenia.

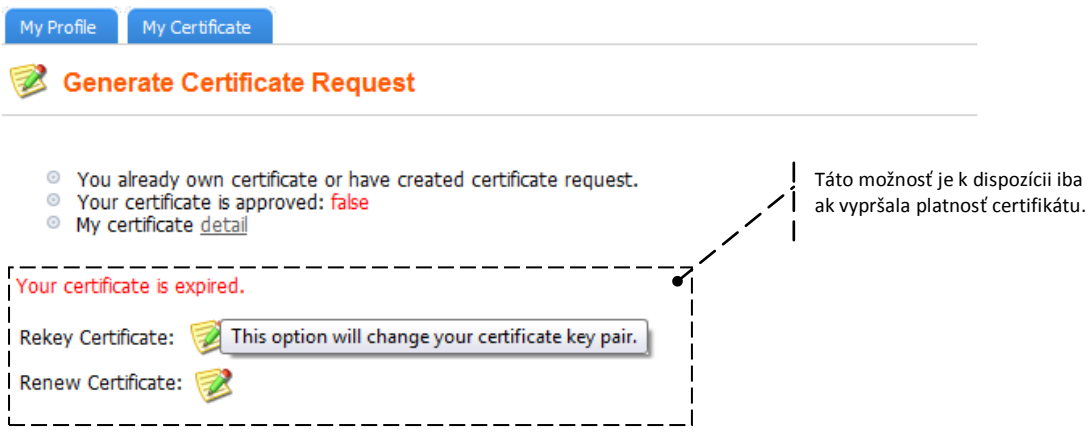
6.8 Obnovenie certifikátu užívateľov

Po vypršaní doby platnosti certifikátu užívateľa sa jeho certifikát automaticky dostáva okamžite (bez odkladu) na zoznam zneplatnených certifikátov - CRL. V prípade ak má užívateľ záujem o obnovu svojho zneplatneného certifikátu, má dve možnosti:

Obnovenie certifikátu rovnakého verejného kľúča (technika Renew) - nový certifikát sa líši od pôvodného v zmenenom sériovom čísle, dobou platnosti avšak verejný kľuč aj predmet certifikátu sú nezmenené v tomto prípade.

Obnovenie certifikátu s vygenerovaním nových párových dát (technika Rekey) - v tomto prípade bude certifikát navyše obsahovať odlišný verejný kľuč od pôvodného.

V systéme sú umožnené obe možnosti obnovy certifikátov a ich implementácia je zobrazená na obrázku 6.11 v sekcii *My Certificate*.



Obr.6.11: Žiadosť o obnovu certifikátu pomocou Renew alebo Rekey .

6.9 Doba platnosti certifikátu CA a certifikátov užívateľov

K jedným z najdôležitejších rozhodnutí poskytovateľa CA je stanovenie dĺžky platnosti jeho certifikátu. Dĺžka certifikátu by mohla byť stanovená za čo najdlhšiu, čím by sme sa vyhli obnove certifikátu CA a všetkému tomu, čo táto operácia zahŕňa. Bezpečnejšie by sa mohlo zdať stanoviť túto dobu na čo najkratšiu, avšak opak je pravdou. Z tohto dôvodu je vhodné voliť dobu platnosti certifikátu CA pomocou akéhosi kompromisu. Keďže neexistuje žiadne oficiálne odporučenie, bola stanovená doba platnosti certifikátu CA na dva roky. Naopak doba platnosti certifikátov samotných užívateľov v systéme sa bežne stanovuje na jeden alebo dva roky. V tomto prípade, bola taktiež stanovená na dva roky.

6.10 Organizácia použitých kódov

Všetky zdrojové kódy (funkcie, metódy) webovej aplikácie sú prehľadne zoskupené do jednotlivých tried, ktorých názvy odpovedajú oblasti ich použitia. Stručný popis jednotlivých tried je možné pochopiť z nasledujúceho popisu:

`Namespace CATHelper` obsahuje:

`public static class User:`
Trieda obsahujúca metódy pre prácu s užívateľskými účtami.

`public class UserInfo:`
Trieda slúžiaca pre vytvorenie objektu samotného užívateľa so všetkými jeho preddefinovanými položkami.

`public static class Context:`
Trieda pre prístup ku kontextovým dátam aktuálneho užívateľa. Každému užívateľovi v systéme sú po úspešnom prihlásení uložené dáta pomocou aktuálnej relácie práve do tejto triedy. Aby nevznikala potreba dotazovať sa do databáze zakaždým, kedy je potreba v aplikácii využívať užívateľské informácie, je práve táto trieda prevažne využívaná pre prácu s dátami aktuálneho užívateľa.

`public static class Settings:`
Trieda obsahujúca metódy pre prácu s nastaveniami aplikácie.

`public static class Timestamp:`
Trieda súžiaca pre prácu s časovým razítkom.

`public static class X509Certificate:`
Trieda obsahujúca metódy pre prácu s certifikátmi a digitálnym podpisom.

`public class Hash:`
Trieda slúžiaca pre generovanie a overovanie hashu dokumentov, globálne dát.

`Namespace EmailEngine` obsahuje:

`public class EmailHelper:`
Trieda súžiaca pre posielanie e-mailových správ.

`public class EmailMessage:`
Trieda slúžiaca pre vytvorenie objektu samotnej e-mailovej správy.

Stručný popis jednotlivých funkcií, tak ako aj ich vstupné a výstupné dátové typy možno prehľadne vidieť v nasledujúcom zozname:

Trieda **`User`**:

`public static DataSet GetUsers(string WhereCondition, string ColumnName_Names):`
Metóda pre získanie vybraných dát všetkých užívateľov aplikácie na základe stanovenej podmienky, ktorej návratový typ tvorí objekt *DataSet*.

`public static void DeleteUser(int userID):`
Metóda pre vymazanie užívateľa z aplikácie na základe jeho identifikačného čísla.

`public static string GeneratePassword(int Lenght):`
Metóda pre generovanie hesla, ktorá ako vstupný parameter poberá dĺžku požadovaného hesla.

`public static bool UserExists(string login):`
Metóda pre zistenie informácie, či už existuje v aplikácii užívateľ so zadaným užívateľským menom.

`public static bool ValidateUser(string login, string password):`
Metóda pre zistenie informácie, či predané užívateľské meno a heslo navzájom korešpondujú.

`public static void ResetPassword(string login, string password):`
Metóda pre nastavenie užívateľského hesla na základe užívateľského mena a nového hesla. Metóda interne volá metódu SetPassword.

`public static void SetUserPassword(string login, string new_password, string existing_password):`
Metóda pre nastavenie užívateľského hesla na základe užívateľského mena starého a nového hesla. Metóda interne volá metódu SetPassword.

`private static void SetPassword(string login, string new_password, string existing_password, bool reset):`
Hlavná metóda, pre nastavenie/resetovanie užívateľského hesla, ktorá je volaná z metód *ResetPassword* a *SetUserPassword*.

Trieda **`Settings`**:

`public static string GetValue(string ColumnName):`
Metóda pre získanie hodnoty z tabuľky *CAT_Settings* zo stĺpca *Value* na základe kódového mena nastavenia.

`public static void SetValue(string ColumnName, string value) :`
Metóda pre zápis hodnoty nastavenia do tabuľky *CAT_Settings* na miesto podľa špecifikovaného kódového mena.

Trieda **`Timestamp`**:

`static public byte[] timestamp(byte[] messageDigest):`
Metóda, ktorá na sa základe vstupného parametra (hashu dokumentu) dotazuje pomocou žiadosti o časové razítko na autoritu časového razítka. Výstupom z tejto funkcie je samotné časové razítko, ktoré môže byť následne uložené do dátového súboru. Táto metóda interne využíva metódu *TimestampRequest*.

`static public HttpWebResponse TimestampRequest(byte[] hash):`
Metóda, ktorá vo svojej podstate vytvára žiadosť o časové razítko.

`static public TimeStampResponse GetTimeStampResponseFromByte(byte[] rawTimeStampResponse):`
Metóda slúžiaca pre opätovné načítanie súboru časového razítka do objektu TimeStampResponse, s ktorým je ďalej možné pracovať.

Trieda **`X509Certificate`**:

`public static void checkCertificates():`
Metóda, ktorá na základe doby platnosti certifikátov nastavuje v databázi príznak každého certifikátu (*isApproved*), na základe ktorého sú certifikáty automaticky presúvané to CRL.

`public static string getKeyXMLString(Guid CertificateGuid):`
Metóda na získanie súkromného kľúča prislúchajúceho certifikátu s *GUID* uvedenom ako vstupný parameter metódy.

`public static bool CertificateIsApproved(int UserID):`
Metóda na zistenie stavu certifikátu na základe *UserID* užívateľa.

`public static bool CertificateIsApproved(Guid cerGUID):`
Metóda na zistenie stavu certifikátu na základe *GUID* certifikátu užívateľa.

`public static bool getCertificateStatus(int userID):`
Metóda na zistenie stavu certifikátu na základe *UserID* užívateľa.

`public static void setCertificateStatus(int UserID, bool value):`
Metóda slúžiaca na nastavenie stavu certifikátu na základe *UserID* užívateľa.

`public static void setCertificateStatus(Guid cerGUID, bool value):`
Metóda slúžiaca na nastavenie stavu certifikátu na základe *GUID* certifikátu užívateľa.

`public static X509Certificate LoadCertificateFromFile(Guid cerGUID):`
Metóda slúžiaca pre načítanie certifikátu zo súborového systému disku na serveri na základe *GUID* certifikátu.

`public static X509Certificate LoadCertificateFromDB(int UserID):`
Metóda slúžiaca pre načítanie certifikátu z databáze na základe *UserID* užívateľa . Metóda interne volá metódu *LoadCertificateFromDBGlobal*.

`public static X509Certificate LoadCertificateFromDB(Guid cerGUID):`
Metóda slúžiaca pre načítanie certifikátu z databáze na základe *GUID* certifikátu užívateľa. Metóda interne volá metódu *LoadCertificateFromDBGlobal*.

`private static X509Certificate LoadCertificateFromDBGlobal(Guid cerGUID, int UserID):`
Hlavná metóda na načitanie certifikátu z databáze volaná pomocnými metódami *LoadCertificateFromDB*.

`private static void WriteCertificateToFile(int UserID, byte[] rawcert, RSA privatekey):`
Metóda slúžiaca pre zápis dát certifikátu na súborový systém disku na serveri a následné uloženie potrebných informácii o certifikáte do databáze. Metóda interne volá metódu *WriteCertificateInfoToDB*.

`private static void WriteCertificateInfoToDB(int UserID, Guid cerGUID, byte[] rawcert, RSA privatekey):`
Metóda na zápis potrebných informácii o certifikáte do databáze, ktorá interne volá metódu *WriteCertificateToDB*.

`private static void WriteCertificateToDB(int UserID, Guid cerGUID, byte[] rawcert, bool writeOnlyCerInfo, RSA privatekey):`
Hlavná metóda na zápis všetkých potrebných informácii o certifikáte do databáze.

`public static void GenerateCert(int UserID, string[] args, bool selfSigned, bool StoreCertInDatabase, RSA HasSubjectKey):`
Metóda slúžiaca na generovanie certifikátu na základe *UserID* užívateľa a ostatných vstupných parametrov.

`public static bool VerifySignature(byte[] data, byte[] signature, Guid SignGuid):`
Metóda na overenie pravosti digitálneho podpisu na základe pôvodných dát, podpisu a *GUID* certifikátu užívateľa.

`public static byte[] SignData(byte[] data, Guid SignGuid):`
Metóda slúžiaca na vytvorenie digitálneho podpisu dát na základe súkromného kľúča *GUID* certifikátu užívateľa.

`public static void DeleteCertFromFileSystem(Guid certGuid):`
Metóda na odstránenie záznamu certifikátu zo súborového systému disku na serveri na základe *GUID* certifikátu.

`public static void DeleteCertFromFileSystem(int UserID):`
Metóda na odstránenie záznamu certifikátu zo súborového systému disku na serveri na základe *UserID* užívateľa.

`public static void DeleteCertFromDatabase(int UserID):`
Metóda na odstránenie certifikátu z databáze na základe *UserID* užívateľa.

`public static void DeleteCertFromDatabase(Guid certGuid):`
Metóda na odstránenie certifikátu z databáze na základe *GUID* certifikátu.

`public static bool isUserPrivateKey(int userID, Guid KeyGuid):`
Metóda na zistenie informácie, či užívateľ ovi na základe jeho *UserID* prislúcha *GUID* certifikátu. Je to z dôvodu potreby overenia či danú užívateľ má prístup k súkromnému kľúču certifikátu špecifikovaného na základe *GUID*.

`public static bool userHasCertificate(int UserID):`
Metóda na zistenie informácie, či užívateľ vlastní v systéme certifikát na základe jeho *UserID*. Metóda interne volá metódu *userHasCertificateGlobal*.

`public static bool userHasCertificate(string UserName):`
Metóda na zistenie informácie, či užívateľ vlastní v systéme certifikát na základe jeho užívateľského mena (login). Metóda interne volá metódu *userHasCertificateGlobal*.

`private static bool userHasCertificateGlobal(string UserName, int userID):`
Hlavná metóda pre zistenie informácie, či užívateľ vlastní v systéme certifikát volaná z metód *userHasCertificate*.

`public static Guid userCertificateGUID(int userID):`
Metóda pre získanie *GUID* identifikátora certifikátu užívateľa na základe jeho *UserID*.

Trieda **`Hash:`**

`public static byte[] ComputeHash(string plainText, string hashAlgorithm):`
Metóda na vytvorenie hashu vstupných dát. Algoritmus hashu je špecifikovaný ako druhý parameter.

`public static byte[] ComputeHash(byte[] plainTextBytes, string hashAlgorithm):`
Metóda na vytvorenie hashu vstupných dát. Algoritmus hashu je špecifikovaný ako druhý parameter.

`public static bool VerifyHash(byte[] plainTextBytes, string hashAlgorithm, byte[] hashValue):`
Metóda pre overenie pravosti hashu dát na základe pôvodných dát, hashu dokumentu a použitého algoritmu.

Trieda **`EmailHelper:`**

`public static void SendEmail(EmailMessage message):`
Metóda pre posielanie e-mailových správ s definovanými hodnotami z nastavení aplikácie.

`public static void SendEmail(EmailMessage message, string server, string username, string password, string encoding, bool useSSL):`
Metóda pre posielanie e-mailových správ, kde sú všetky vstupné parametre potrebné nastaviť. Prvá metóda volá interne druhú s predefinovanými nastaveniami.

7 Záver

Pri tvorbe diplomovej práce som čerpal informácie z internetových zdrojov a z knižnej literatúry venujúcej sa problematike kryptografie, infraštruktúry verejného kľúča a programovacieho jazyka ASP.NET od spoločnosti Microsoft. Tieto témy, nie sú ľahko zmapovateľné, pretože neustále rozvíjajúce sa nové technológie posúvajú tieto odvetvia každým dňom na inú, vyššiu úroveň.

Hlavným cieľom tejto práce bolo popísať princíp fungovania infraštruktúry verejných kľúčov a s tým súvisiaci význam digitálneho certifikátu v tomto kontexte. Boli spomenuté aj najzákladnejšie princípy funkčnosti symetrického, asymetrického šifrovania a jednosmerných hashovacích funkcií, na ktorých je založený aj samotný princíp digitálneho podpisu. Ďalej som sa sústredil na popis v súčasnosti nie veľmi známej techniky overovania existencie platného digitálneho podpisu v určitom časovom momente - techniky časového razítka.

V predposlednej časti som popísal zvolený programovací jazyk, ktorý sa stal základným nástrojom samotnej diplomovej práce. V tejto časti boli popísané najzákladnejšie funkcie pre prácu s digitálnymi certifikátmi, ktoré sú už implementované a sú súčasťou .NET Frameworku.

Ako praktická časť diplomovej práce bola vytvorená webová aplikácia postavená na platforme ASP.NET, ktorá prakticky aplikuje doposiaľ nadobudnuté teoretické znalosti z oblasti digitálneho podpisu a časového razítka do praxe. Aplikácia predstavuje certifikačnú autoritu, ktorá vydáva certifikáty užívateľom systému, ktorý následne môžu digitálne podpisovať dokumenty a overovať pravosť digitálneho podpisu.

Myslím si, že v súčasnej dobe je laická verejnosť ešte veľmi málo oboznámená s princípom digitálneho podpisu v praxi, ktorý významným spôsobom môže uľahčiť komunikáciu medzi obyvateľmi a štátnymi orgánmi alebo rôznymi inými inštitúciami. Z celej práce vyplýva, že najpodstatnejším faktorom celého systému zabezpečenia identity komunikujúcich účastníkov je, pri dodržaní správnych zásad v prípade budovania infraštruktúry verejných kľúčov, nezameniteľnosť identity komunikujúcich účastníkov, ku ktorej nesmie dôjsť, či už z dôvodu útoku z tretej strany alebo neopatrnosti komunikujúcich účastníkov.

Diplomová práca mala pre mňa veľký prínos v podobe nadobudnutia vedomostí z oblasti PKI a taktiež v dosiahnutí vytvorenia zadanej webovej aplikácie, ktorá zahrňuje teoretickú časť tejto práce.

Zoznam použitej literatúry

[1] *Elektronický podpis: Přehled právní úpravy, komentář k prováděcí vyhlášce k zákonu o elektronickém podpisu a výklad základních pojmů. Olomouc : ANAG, 2002. 141 s. ISBN 80-7263-125-X.*

[2] *DOSTÁLEK, Libor, VOHNOUTOVÁ, Marta. Velký průvodce infrastrukturou PKI a technologií elektronického podpisu. 1. vyd. Brno : Computer Press, 2006. 534 s. ISBN 80-251-0828-7.*

[3] *DOSTÁLEK, Libor. Velký průvodce protokoly TCP/IP : Bezpečnost. 2. aktualiz. vyd. Praha: Computer Press, 2003. xvi, 571 s. ISBN 80-7226-849-X.*

[4] *DOSEDĚL, T. Počítačová bezpečnost a ochrana dat. Brno: Computer Press, 2004. 190 s. ISBN 80-251-0106-1.*

[5] *Microsoft Developer Network. 2008. [online]. [cit. 7.12.2008]. Dostupné z URL: < http://msdn.microsoft.com/>.*

[6] *Network Working Group. RFC3161 - Internet X.509 Public Key Infrastructure Time-Stamp P. 2001. [online]. [cit. 7.12.2008]. Dostupné z URL: <http://www.faqs.org/rfcs/rfc3161.html>.*

[7] *Network Working Group. RFC3280* - Policy Requirements for Time-Stamping Authorities (TSAs)*. 2003. [online]. [cit.7.12.2008]. Dostupné z URL: <http://www.ietf.org/rfc/rfc3280.txt>.*

[8] *Network Working Group. RFC3628 - Internet X.509 Public Key Infrastructure* Certificate and Certificate Revocation List (CRL) Profile*. 2002. [online]. [cit.7.12.2008]. Dostupné z URL: <http://www.ietf.org/rfc/rfc3628.txt>.*

Zoznam použitých skratiek

ANS.1	Abstract Syntax Notation One
ASP.NET	Active Server Pages .NET (Aktívne serverové stránky .NET)
BER	Basic Encoding Rules (Základné pravidlá pre kódovanie)
CA	Certificate Authority (Certifikačná Autorita)
CAT	Certificate Authority and Timestamping
CLR	Common Language Runtime (Behový jazyk)
CRL	Certificate Revocation List (Zoznam odvolaných certifikátov)
CTL	Certificate Trusted List (Zoznam dôveryhodných certifikátov)
DSA	Digital Signature Algorithm (Algoritmus digitálneho podpisu)
HTML	HyperText Markup Language (Hyper-textový znakový jazyk)
MD	Message Digest (Hash správy)
MIME	Multipurpose Internet Mail Extensions (Viacúčelové rozšírenia e-mailu)
MSIL	Microsoft Intermediate Language (Medzilahlý jazyk Microsoftu)
OID	Object Indentifier (Identifikátor objektu)
OSCP	On-line Certificate Status Protocol (On-line protokol pre zistenie stavu certifikátu)
PKI	Public Key Infrastructure (Infraštruktúra verejného kľúča)
RSA	Rivest, Shamir, Adleman
TS	Time Stamp (Časové razítko)
TSP	Time-Stamp Protocol
TSA	Time Stamp Authority (Autorita časových razítok)
UTC	Coordinated Universal Time (Univerzálne koordinovaný čas)

Príloha A

Tab.2: Rozšírené príkazy nástroja Makecert.

Príkaz	Popis príkazu
-a algorithm	Špecifikuje algoritmus podpisu. Môže to byť **md5** alebo **sha1**.
-b mm/dd/yyyy	Špecifikuje počiatočný dátum platnosti certifikátu.
-cy certType	Špecifikuje typ certifikátu. Môže nadobudnúť hodnôt **end** alebo **authority**.
-d name	Vypíše meno predmetu.
-e mm/dd/yyyy	Špecifikuje koncový dátum platnosti certifikátu. Pokiaľ nie je špecifikovaný, je nastavený na 12/31/2039 11:59:59 GMT.
-eku oid[,oid]	Vkladá zoznam pomlčkou-oddelených identifikátorov použitia kľúča do certifikátu.
-h number	Špecifikuje maximálny level stromu, ktorý sa môže nachádzať pod týmto certifikátom.
-ic file	Špecifikuje certifikačný kľuč vydavateľa certifikátu.
-ik keyName	Špecifikuje meno kľúča vydavateľa.
-iky keytype	Špecifikuje typ kľúča vydavateľa, ktorý musí byť **signature** alebo **exchange**.
-in name	Špecifikuje úplné meno certifikátu vydavateľa.
-ip provider	Špecifikuje vydavateľovo meno poskytovateľa CryptoAPI.
-ir location	Špecifikuje umiestnenie certifikačného úložišťa vydavateľa. Môže byť **currentuser** alebo **localmachine**.
-is store	Špecifikuje meno certifikačného úložišťa vydavateľa.
-iv pvkFile	Špecifikuje súkromný kľuč .pvk certifikačného súboru vydavateľa certifikátu.
-iy pvkFile	Špecifikuje typ CryptoAPI poskytovateľa vydavateľa.
-l link	Odkazuje na doplnkové informácie (napríklad URL).
-m number	Špecifikuje dĺžku, v mesiacoch , platnosti certifikátu.
-nscp	Pripája rozšírenie klientskej autentizácie Netscape.
-r	Vytvára koreňový certifikát.
-sc file	Špecifikuje certifikačný súbor predmetu certifikátu.
-sky keytype	Špecifikuje typ kľúča predmetu certifikátu, ktorý musí byť **signature**, **exchange**, alebo číslo reprezentujúce typ poskytovateľa.
-sp provider	Špecifikuje meno CryptoAPI poskytovateľa predmetu certifikátu.
-sv pvkFile	Špecifikuje súkromný kľuč predmetu v .pvk súbore.
-sy type	Špecifikuje typ CryptoAPI poskytovateľa predmetu.