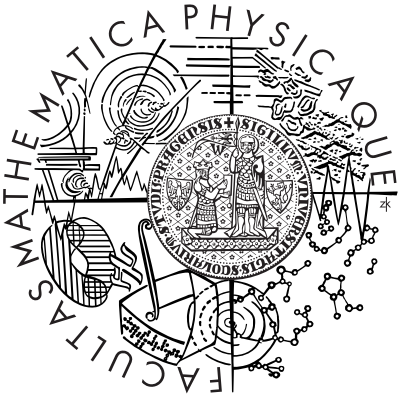
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Tomáš Vrábel

Implementace dlouhodobého elektronického podpisu

Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Libor Dostálek
Studijní program: Informatika, Softwarové systémy

2008

Ďakujem vedúcemu diplomovej práce RNDr. Liborovi Dostálkovi za ochotu a pomoc pri jej tvorbe a tiež svojím rodičom, ktorí mi umožnili študovať a celé roky ma podporovali.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 8.8.2008 Tomáš Vrábel

Obsah

Názov práce: Implementace dlouhodobého elektronického podpisu
Autor: Tomáš Vrábel
Katedra (ústav): Katedra softwarového inženýrství
Vedúci diplomovej práce: RNDr. Libor Dostálek
e-mail vedúceho: libor.dostalek@siemens.com

Abstrakt: Pre elektronický podpis existujú dva štandardy. Ani jeden však nerieši platnosť podpisu v čase a dlhodobú archiváciu podpisu. Túto problematiku rieši až štandard CAdES vydaný ako štandard ETSI TS 101 733 alebo RFC 5126. V súčasnosti však neexistuje uspokojivá implementácia tohoto štandardu. Cieľom tejto diplomovej práce bolo práve analyzovať a implementovať štandard CAdES nad open-source knižnicou OpenSSL. V úvode práca popisuje infraštrutúru PKI a problematiku digitálneho popisu. Práca ďalej analyzuje štandard CAdES a analyzuje možnosti jeho implemetácie. Nakoniec sa práca zaoberá samotným spôsobom implemtácia štandardu do OpenSSL ako aj možnosťami ďalšieho rozvoja. Súčasťou implementácie nie je len súbor funkcií, ale aj riadková utilita, ktorá umožňuje vytvárať a overovať podpis podľa štandardu CAdES. Samotná implementácia bola tvorená so zreteľom na prípadne zarhnutie implementácie do oficiálnej distribúcie OpenSSL.
Kľúčové slová: CAdES, dlhodobý elektronický podpis, implementácia, RFC 5126

Title: Long term signature implementation
Author: Tomáš Vrábel
Department: Department of Software Engineering
Supervisor: RNDr. Libor Dostálek
Supervisor's e-mail address: libor.dostalek@siemens.com

Abstract: Today there exist two standards for electronic sigatures. None of them is discussing validity of signature in long period of time. This problem is solved by CAdES standard, described in ETSI TS 101 733 or RFC 5126. Presenty there is no satisfying implementation of CAdES standard. The aim of this diploma work is to analyse and implement CAdES standard using open-source library OpenSSL. The work begins with introduction of PKI standards and problem of electronic signatures. The work continues with analyses of CAdES standard and analyses of the implementation of CAdES standard. Finally, this work is describing the implementation of standard CAdES using OpenSSL library and also possiblities of further development. A part of the implementation is a command line utility that allows users to create and verify long term signatures according to CAdES standard easily. The implementation was created with respect to possibility of extending current OpenSSL distribution.
Keywords: CAdES, long term signature, implementation, RFC 5126

Kapitola 1

Úvod

S rozšírením počítačov do mnohých sfér ľudskej činností stúpa aj množstvo dát v elektronickej forme. Pri požiadavke na autenticitu a integritu dát sa používa elektronický podpis. Jeho súčasnému rozšíreniu určite pomohli aj legislatívne úpravy, ktoré elektronický podpis postavili na úroveň rukou vytvoreného podpisu. Pre elektronický podpis existujú dva štandardy:

- Cryptographic Message Syntax (CMS - predtým tiež označovaný ako PKCS#7) [1]. Tento štandard špecifikuje formát nielen elektronicky podpísaných správ, ale aj autentizovaných správ, elektronickej obálky (tj. šifrovaných správ), atď.

- XMLsignature [4], ktorý využíva syntax XML. Tento typ elektronického podpisu prináša mnoho výhod pre podpisovanie dokumentov, ktoré samotné majú XML štruktúru.

Oba štandardy (CMS aj XMLsignature) však riešia iba elektronický podpis v okamihu jeho vzniku, tj. neriešia platnosť elektronického podpisu v čase. Pritom pre archiváciu elektronicky podpísaných dokumentov je potrebné udržiavať platnosť elektronického podpisu v čase.

K obom štandardom (CMS aj XMLsignature) preto existuje rozšírenie riešiace konzerváciu elektronického podpisu v čase:

- K štandardu CMS existuje rozšírenie CAdES, ktoré bolo štandardizované IETF ako RFC 5126 [3] a zároveň štandardizované ETSI pod označením ETSI TS 101 733 [2].

- K XMLsignature existuje obdobné rozšírenie XAdES [4].

Pre implementáciu štandardu CMS sa všeobecne používa OpenSSL, avšak ani OpenSSL neimplementuje rozšírenie CAdES podľa štandardu ETSI TS 101 733. Cieľom tejto diplomovej práce je práve implementácia CAdES podľa štandardu ETSI TS 101 733.

1.1 Presné vymedzenie cieľov

Cieľom je podľa štandardu ETSI TS 101 vytvoriť:

- Knižnicu funkcií splňujúcu nasledovné body:
  1. Implementácia bude vychádzať z OpenSSL, tj. aby implementácia bola ľahko využiteľná vývojármi majúcimi skúsenosti s používaním OpenSSL
  2. Implementácia bude implementovať celý štandard CAdES, okrem podpisu typu EPES (doteraz existujúce implementácie implementujú vždy len jednu z vlastností CAdES).
  3. Implementácia bude naviac podporovať verifikáciu elektronického podpisu k zadanému času v minulosti, kedy elektronický podpis potencionálne vznikol. Táto vlastnosť nie je súčasťou štandardu CAdES, ale je logickým požiadavkom budúcich aplikácií (napr. aplikácií zaručených elektronických archívov)
- Riadkovú utlitu, ktorá bude poskytovať všetky základné funkčnosti knižnice.

Celý zdrojový kód bude napísaný v jazyku C, nakoľko aj samotná OpenSSL je napísaná v tomto jazyku.

1.2 Popis problematiky, význam môjho riešenia

Doba platnosti elektronického podpisu je určená platnosťou podpisového certifikátu. Platnosť takýchto certifikátov je zvyčajne niekoľko mesiacov až rokov (záleží na certifikačnej autorite), po skončení platnosti podpisového certifikátu je podpis neplatný. Hľadajú sa preto riešenia, ako dokázať autenticitu podpisu aj v dlhodobom časovom horizonte [1].

Priamočiare riešene problému s dĺžkou platnosti el. podpisu predstavuje vydanie podpisového certifikátu na požadované obdobie. Slabinou tohoto riešenia je, že bezpečnosť súkromeho kľúča sa nedá garantovať na ľubovolné obdobie a súkromný podpisový kľúč by mohol byť pri dlhej platnosti certifikátu kompromitovaný. Takéto riešenie tiež nie je dostatočne škálovatelňé v dobe platnosti podpisu. Požiadavka na platnosť podpisu, ktorá má byť dlhšia ako súčasne platný podpisový certifikát by si vyžiadala vydanie certfikátu s požadovanou platnosťou. Platnosť certifikátov je však v kompetencií certifikačných autorít, certifikát s platnosťou niekoľko desaťročí však v súčasnosti nevydá žiadna certifikačná autorita. Perspektívne riešenie spočíva

[1] Platnosť elektronického podpisu na dlhé obodbie sa vyžaduje napríklad pri archivovaní dokumentov

vo využití časových razítok ako dôkazu existencie podpisu v minulosti. K elektronickému podpisu sa doplní časové razítko, ktoré potrvrdzuje, že digitálny podpis existoval v čase vydania časového razítka. Po skočení platnosti podpisového certfikátu existuje platné časové razítko, ktoré dokazuje, že digitálny podpis existoval v dobe, keď bol podpisový certifikát platný. Podpis preto nemohol byť vytvorený na základe kompromitovaného súkromného kľúča podpisového certifikátu.

Ani časové razítko však nemá neomedzenú platnosť. Časové razítko je digitálne podpísaná štruktúra a po skončení platnosti podpisového certifikátu časovej autority razítko stratí platnosť. Riešením je tzv. predlžovanie platnosti digitálneho podpisu:

Určité časti podpisu spolu s pôvodným časovým razítkom sa ešte pred skončením platnosti pôvodného časového razítka orazítkujú novým časovým razítkom. Nové časové razítko dokazuje, že platný podpis existoval v dobe vydania časového razítka. Tým predĺžia jeho platnosť, kým neskončí platnosť nového časového razítka. Časové razítko tak predĺži platnosť podpisu do doby skončenia platností časového razítka.

Kapitola 2

Infraštruktúra PKI

PKI alebo Public Key Infrastructure popisuje mechanizmus dohody predtým neznámych strán (užívateľov, počítačov) a ich vzájomnú autentizáciu. Mechanizmus je založený na princípe asymetrickej kryptografie a popisuje štruktúry a protokoly, ktoré vzájomnú autentizáciu umožnia. PKI ďalej popisuje spôsoby zaručenia integrity a nepopierateľnosti správ, oprávnení, dôkazy o existencií dokumentu v čase a podobne. V nasledovnej časti budú popísané základné štruktúry a protokoly PKI a spôsoby ich použitia. Štandard CAdES totiž vychádza z mnohých štandardov PKI a pri implementácií štandardu CAdES bolo nutné sa s týmito štandardami zoznámiť. Medzi najdôležitejšie štandardy patria štandardy popisujúce certifikáty, CRL, digitálny podpisu a časové razítko.

2.1 Asynchrónna kryptografia

Asynchrónna kryptografia je typ kryptografie, kde je namiesto jedného kľúča použitého na šifrovanie aj dešifrovanie, použitá dvojica kľúčov. Jednému hovoríme súkromný kľúč a druhému verejný. Dáta zašifrované jedným z kľúčov, musia byť dešifrované druhým z dvojice kľúčov. Korektnosť operácií šifrovanie a dešifrovanie vychádza z matematických poznatkov a podľa toho sa rozlišujú príslušné asymetrické šifrovacie algoritmy (RSA,DSA,Diffle-Hellman...). Použiteľnosť týchto algoritmov v kryptografii vychádza z toho, že zatiaľ nie je známy obecný deterministický polynomiálny algortimus na prevod verejného kľúča na privátny. Existujú však metódy, ktoré dokážu pri dostatočnej výpočetnej sile privátny kľúč odhaliť. Je preto nutné počítať s kompromitáciou súkromného kľúča a dvojicu kľúčov vo vhodne zvolených časových intervaloch obnovovať.

Asymetrická kryptografia sa používa v týchto prípadoch:

- šifrovanie verejným kľúčom - správa zašifrovaná verejným kľúčom príjmateľa nemôže byť dešifrovaná nikým iným, iba vlastníkom príslušného privátneho kľúča

- digitálny podpis - správa zašifrovaná vlastníkom privátneho kľúča môže byť overená pomocou verejného kľúča. Zašifrovanie správy sa potom považuje za podpis správy. Overenie pomocou verejného kľúča zaručuje autenticitu aj integritu pôvodnej správy.

- Bezpečné vytvorenie zdieľaného kryptografického tajomstva (napr. symetrického šifrovacieho kľúča). K tomuto účelu slúži algoritmus Diffle-Hellman.

Problém spojený s použitím verejného kľúča je dôveryhodnosť takéhoto kľúča. Ak je verejný kľúč distribuovaný nedôveryhodným kanálom, môže byť podvrhnutý. Je preto žiadúce spojiť verejný kľúč s indentitou jeho držiteľa. Je tiež žiadúce mať dôkaz, že osoba vlastní k verejnému kľúču, ktorý distribuuje, aj príslušný privátny kľúč. Tieto problémy rieši tzv. certifikácia verejného kľúča. Špeciálny orgán (nezávislá tretia strana) na základe žiadosti užívateľa vytvorí certifikát, v ktorom k verejnému kľúču užívateľa pripojí identifikáciu užívateľa, a toto spojenie údajov podpíše svojim súkromným kľúčom. Tento orgán tvorí väčšinou nezávislá tretia strana, ktorá má dôveru všetkých, ktorí budú s ňou certifikovanými kľúčmi pracovať. Svojim podpisom sa orgán zaručí, že dáta uvedené v certifikáte sú pravdivé. Tento orgán sa nazýva certifikačná autorita.

Verejný kľúč certifikačnej autority, ktorý je potrebný na overenie certifikátu užívateľa, je tiež certifikovaný. Certifikát svojho verejného kľúča si mohla vydať certifikačná autorita sama, prípadne jej ho na základe žiadosti vydala iná certifikačná autorita.

2.2 Certifikát verejného kľúča

Certifikát verejného kľúča je základom infraštruktúry PKI, spája identitu držiteľa (meno, adresu) verejného kľúča s príslušným verejným kľúčom. Toto spojenie je podpísané certifikačnou autoritou, podpisom sa za správnosť údajov uvedených v certifikáte certifikačná autorita zaručuje. V tejto kapitole bude podrobnejšie popísaný certifikát verejného kľúča, jeho životný cyklus a spôsob overenie platnosti. Certifikát verejného kľúča definuje niekoľko štandardov. Pre potreby Internetu je definovaný certifikát v štandarde X.509, ktorý vydal ITU. Tento štandard bol rozširovaný a v súčasnosti je platný RFC-5280 [10].

Dátová štuktúra certifikátu obsahuje mimo iné tieto položky:

- verzia certifikátu - tento údaj určuje verziu certifikátu. Časom sa štandard certifikátu rozširoval a vznikali nové verzie, v súčasnosti je platná verzia 3 a iné sa nepoužívajú.

- sériové číslo certifikátu - celé kladné číslo. Musí jedinečné pre každý certifikát vydaný certifikačnou autoritou. Tento údaj sa používa na

jednoznačnú identifikáciu certifikátu v rámci všetkých certifikátov vydaných certifikačnou autoritou.

- algoritmus podpisu - dvojica algoritmov (algoritmus otlačku a šifrovania) použitá pri výpočte podpisu certifikátu
- platosť - položka určuje platnosť certifikátu. Platnosť certifikátu udáva interval ohraničený časmi Platosť od (Not Before) a Platnosť do (Not After)
- vydavateľ - položka identifikuje vydavateľa certifikátu
- predmet - položka identifikuje držiteľa certifikátu
- verejný kľúč - položku tvori dvojica informácií: identifikácia algoritmus, pre ktorý je verejný kľúč určený a príslušný verejný kľúč
- rozšírenia certifikátu - upresňujúce a doplňujúce údaje

Táto štruktúra je podpísaná súkromným kľúčom certifikačnej autority a tento podpis potom pridaný ku štruktúre certifikovaných údajov [1]. Certifikované údaje spolu s podpisom tvoria certifikát.

Platnosť certifikátu je udaná dvoma časmi - Platnosť Od určuje čas, od kedy je certifikát platný, Platnosť Do určuje čas, po ktorom je certifikát neplatný. Každá certifikačná autorita pri určovaní plastnosti certifikátov zohľadňuje mnoho kritérií. Ak je platnosť certifikátu krátka, certifikát je nutné často obnovovať. Ak je však platnosť certifikátu veľmi dlhá, hrozí, že súkromný kľúč, ktorý certifikát certifikuje, bude odhalený.

Platnosť certifikátu vydavateľa (certifikačnej autority) by mala vždy presahovať platnosť ňou vydaných certifikátov. Certifikát vydavateľa (certifikačnej autority) musí byť totiž platný, aby bol vydaný certifikát dôveryhodný[2]. Ak má byť certifikát certifikačnej autority platný menej ako N dní, nemala by vydať certifikát s platnosťou N dní na zákade tohoto certifikátu. Certifikačná autorita v takomto prípade vygeneruje novú dvojici verejný/súkromný kľúč, certifikuje verejný klíč, a nové certifikáty vydáva na základe tohoto nového certifikátu. V tomto obodobí existujú dva platné certifikáty certifikačnej autority s rozdielnymi verejnými kľúčmi, identifikácia certifikátu na základe vydavateľa preto nestačí. Boli preto zavedené rozšírenia Identifikácia kľúča predmetu (Subject Key Identifier) a Identifikácia kľúča autority (Authority Key Identifier). Certifikačná autorita spočíta hash napr. otlačok zo svojho verejného kľúča a vloží ho (alebo jeho časť) do svojho certfikátu ako rozširujúci atribút pod typom Identifikácia kľúča predmetu. Vydaným certifikátom potom certifikačná autorita vkladá do certifikátu túto istú hodnotu (hash

[1] Podpis sa nespočíta z celej štruktúry certifikovaných údajov, ale z hash otlačku tejto štuktúry.

[2] Ak je certifikát vydavateľa neplatný, môže byť súkromný kľúč vydavateľa známy, a ním vydané certifikáty podvrhnuté.

otlačok z verejného kľúča), typ pridaného atribútu je však Identifikácia kľúča úradu. Overovateľ potom na základe identifikácie verejného kľúča úradu vyberie príslušný certifikát certifikačnej autority.

Rozšírenia certifikátu umožňujú pridať ďalšie dodatočné informácie o certfikáte. Rozšírenie certifikátu tvorí zoznam atribútov, jednotlivé atribúty sú zložené z informácie o type atribútu a hodnoty atribútu. Do certifikátu je možné pridať ľubovolné množstvo atribútov. Rozšírenie pridané do certifikátu vo forme atribútu, môže byť označené ako závažné (critical), pri overovaní certifikátu potom nemôže byť takéto závažné rozšírenie ignorované a mechanizmus na overovanie certifikátov tomu závažnému rozšíreniu musí rozumieť. Z pohľadu tejto práce sú zaujímavé tieto rozšírenia:

- Identifikácia kľúča úradu - používa sa na identifikáciu certifikátu podľa verejného kľúča v prípade, že vydavateľ drží viacero certifikátov.
- Identifikácia kľúča predmetu - identifikuje verejný kľúč certifikátu, väčšinou na základe jeho hash otlačku
- Použitie kľúča - toto rozšírenie umožňuje obmedziť použitie certifikovaného verejného kľúča, tj. obmedziť použitie certifikátu.
- Rozšírené použitie kľúča - umožňuje pridávať ďalšie typy použitia certifikátu
- Základné obmedzenia - rozširenie používané v certifikátoch certifikačných autorít. Umožňuje určiť, či sa jedná o certifikát certifikačnej autority a tiež počet podriadených certifikačných autorít[3].

2.3 Certifikačná politika, politika elektronického podpisu

Štandardy popisujúce certifikáty a elektronické podpisy určujú postupy, podľa ktorých je certifikát, resp. digitálny podpis overovaný (verifikovaný). Nestanovujú však množstvo detailov (platnosť certifikátov, CRL, právne záležitosti), ktoré sú pri overovaní certifikátu tiež dôležité. Práve tieto upresnenia obsahuje certifikačná politika.

Certifikačná politika je dokument, ktorý určuje postupy, praktiky a ciele slúžiace k overeniu certifikátu predtým, ako je použitý. Popisuje pravidlá, ktorými sa riadia jednotliví aktéri procesu verifikácie - certifikačná autorita, držiteľ certifikátu a užívateľ certifikátu. Popisuje tiež pravidlá, za akých vydáva certifikačná autorita ceritifikáty, a akým spôsobom za ne autorita ručí. Tieto pravidlá sú popísané v dokumente Certifikačná politika, ktorý vydá certifikačná autorita. Jedná sa o verejný dokument, ktorý je prístupný na

[3]Toto rozšírenie musí byť nastavená ako závažné a certifikát certifikačnej autority ho musí obsahovať.

Internete, a odkaz na tento dokument môže byť vložený do certifikátu v rozšírení Certifikačné politiky.

Obdoba certifikačnej politiky existuje aj pre elektronické podpisy. Ak je elektronický podpis vydaný podľa politiky elektronického podpisu, podpisovateľ a overovateľ musí podľa tejto politiky pri tvorbe resp. overovaní elektronického podpisu postupovať. Štandard CAdES umožňuje do elektronického podpisu vložiť referenciu na tatúto politiku. Politika môže byť vydaná či už ako dokument, alebo v štandardizovanej forme, aby mohla byť strojovo overená.

2.4 Atribútové certifikáty

Atribútové certifikáty umožňujú spojiť atribúty (vlastnosti) a konkrétnou identitou držiteľa a toto spojenie certifikovať. Podpisom atribútová autorita zaručuje, že držiteľ certifikátu má atribúty uvedené v atribútovom certfikáte. Atribúty môžu mať rôznu podobu (meno, adresa, autorizačné oprávnenia, dátum narodenia..) a môžu byť šifrované.[4]

Špecifikovať držiteľa atribútového možno viacerými spôsobmi. Všetky tieto spôsoby však predpokladajú, že držiteľ atribútového certifikátu vlastní aj certifikát verejného kľúča. Certifikát verejného kľúča (a tým aj držiteľ certifikátu) je potom identifikovaný v atribútovom certifikáte aspoň jednou z možností:

- na základe vydavateľa certifikátu a sériového číslo certifikátu
- použije sa jedinečné meno držiteľa certfikátu, uvedené v certifikáte v poli Predmet
- na základe hash otlačku certfikátu verejného kľúča, prípadne samotného verejného kľúča

Atribútové certifikáty sa často používajú pri certifikovaní presnosti hodín autority pre výdaj časových razítok. Časová autorita podpisuje časové razítka súkromným kľúčom, ku ktorému je vydaný certifikát. Po vyhodnotení presnosti hodín časovej autority vydá dôveryhodná autorita atribútový certifikát, ktorý certifikuje presnosť hodín časovej autority. Časová autorita je potom v atribútovom certifikáte identifkovaná napr. na základe mena držiteľa spolu s otlačkom z certifikátu. Takýto atribútový certifikát, ktorý certifikuje presnosť hodín časovej autority používa aj První certifikáční autorita (ICA). Atribútový certifikát má v prípade ICA platnosť tri dni. Takéto atribútové certifikáty môžu byť zahrnuté do štruktúry digitálneho podpisu rovnako ako certifikáty verejného kľúča a sú potom distribuované spolu s digitálnym podpisom.

[4]Certifikát verejného kľúča síce tiež môže obsahovať dodatočné informácie o vlastníkovy certifikátu, na vkladanie určitých typov atribútov však nie je vhodný.

2.5 Odvolávanie certifikátu

Vzhľadom na to, že certifikáty verejného kľúča majú pomerne dlhú dobu platnosti, pravdepodobnosť, že privátny kľúč bude kompromitovaný, nie je zanedbateľná. [5] Bol preto vytvorený mechanizmus na odvolávanie certifikátov, ktorým je možné certifikát zneplatniť ešte pred skončením platnosti certifikátu a tým zabrániť zneužitiu privátneho kľúča. Požiadavku na zneplatnenie certifikátu môže držiteľ kompromitovaného súkromného kľúča oznámiť rôznymi spôsobmi - telefonicky, cez internetový portál certifikačnej autority alebo osobne. Certifikačná autorita informácie o zneplatnených certifikátoch distribuuje dvoma spôsobmi - vo forme CRL, alebo tzv. OCSP protokolom.

CRL je presne popísané v štandarde RFC 2580 [10]. CRL (Certificate Revocation List) alebo zoznam zneplatnených certifikátov je štuktúra obsahujúca informácie o vydavateľovi CRL, dátume vydania a zoznam zneplatnených certifikátov. Zneplatnené certifikáty v zozname sú identifikované podľa sériového čísla certifikátu. Ku sériovému číslu môže byť pridaný aj čas obdržania požiadavky na zneplatnenie certifikátu.[6]

Celá štruktúra CRL je podpísaná privátnym kľúčom vydavateľa CRL (zpravidla certifikačnej autority) a umiestnená na distribučné miesto, odkiaľ ju môžu získať užívatelia. Podpis CRL certifikačnou autoritou poskytuje dôkaz dôveryhodnosti. Zneplatnený certifikát je uverejňovaný na CRL až do času skončenia platnosti certifikátu. Vtedy by bol certifikát vyhlásený za neplatný aj na základe doby jeho platnosti a teda ho nie je ďalej nutné zverejňovať na CRL. Podobný mechanizmus ako pre zneplatňovanie certifikátov verejného kľúča funguje aj pre atribútové certifikáty. Štruktúra obsahujúca odvolané atribútové certifikáty sa nazýca ACRL -Attribute Certificate Revocation List.

CLR by malo byť vydávané podle pravidiel uvedených v certifikačnej politike. Aktuálnosť CRL je veľmi dôležitá, CRL obsahuje len certifikáty odvolané v dobe vydania CRL. Ak je certifikát platný podľa CRL starého niekoľko dní, existuje oprávnená pochybnosť, či v priebehu týchto niekoľkých dní nebol certifikát odvolaný. Certifikačné autority preto vydávajú CRL aj niekoľko krát denne, niektoré ich dokonca vydávajú po každej požiadavke na zneplatnenie certifikátu. CRL obsahuje informáciu o čase vydania ďalšieho CRL, pôvodné CRL je po tomto čase neplatné a je nutné si zaobstarať aktuálne CRL.[7] Platnosť CRL je takto zhora ohraničená časom vydania ďalšieho CRL (resp. časom vydania ďalšieho CRL uvedeným v akuálnom CRL), zaujímavejšie je dolné časové ohraničenie CRL. Dolné ohraničenie

[5] Privátny kľúč môže byť kompromitováný napríklad naopatrným zaobchádzaním, čim môže dôjsť k jeho vyzradeniu.

[6] Čas obdržania požiadavky na zneplatnenie je len informačný údaj, certifikát totiž nie je neplatný od doby zaslania požiadavky na zneplatnenie alebo až od doby uverejnenia certifikátu na CRL.

[7] Certifikačná autorita však môže vydať nové CRL aj pred časem uvedeným v predchádzajúcom CRL (a obecne tak aj koná), je preto v záujme užívateľov, aby si pravidelne CRL aktualizovali.

platnosti CRL je nutné diskutovať v dvoch prípadoch:

1. Certifikát platný podľa aktuálneho CRL bol určite platný aj pred dobou vydania tohoto CRL. Informáciu o platnosti je teda možné aplikovať aj na čas pred vydaním CRL (na CRL vydaných v minulosti určite certifkt uverejnený nebol) a platnosť CRL v prípade neodvolaných certifikátov nie je zdola ohraničená.

2. Ak je certifikát neplatný podľa aktuálneho CRL, nie je možné povedať o jeho platnosti pred dobou vydania CRL nič. Záznam o zneplatnení síce obsahuje údaj o čase obdržania požiadavky na zneplatnenie, tento údaj je však len orientačný. Certifikát mohol byť uverejnený aj starších CRL, toto však nie je isté a preto nie je možné o platnosti certifikátu pred vydaním CRL nič konštatovať.

Celá diskusia o platnosti certifikátu pred dobou vydania CRL prepokladá, že ak je certifikát uverejnený na CRL a teda je neplatný, nie je možné ho už z CRL vyňať (vynimkou je mechanizmus pozastavenia platnosti certifikátu).

Je žiadúce mať dôkaz o platnosti certifikátu v dobe jeho použitia. Nie je možné túto platnosť bez pochýb dokázať aktuálnym CRL. Je lepšie vziať novšie CRL, u ktorého je isté, že ak by bol certifikát v dobe použitia neplatný, na takomto CRL by už bol uverejnený. Takéto CRL je pri procese overovania relevantné. Relevatné CRL je určené aktuálnosťou CRL z certifikáčnej politiky, vždy je to však CRL vydané po čase použita súkromného kľúča - v dobe použita súkromného kľúča toto CRL ešte určite nie je dostupné. Ak je certifikát podľa relevantného CRL platný, nie je možné pochybovať o platnosti certifikátu a súkromného kľúča v čase jeho použita.

Mechanizmus CRL má problémy so škálovateľnosťou, dostupnosťou a administratívnou náročnosťou vydávania CRL. CRL podľa pôvodných špecifikácií obsahovalo zoznam zneplatnených certifikátov. Veľkosť CRL tak bola závislá na počte vydaných certifikátov, dobe platnosti certifikátu a pravdepodobnosti zneplatnenia certifikátu. Veľkosť jedného dokumentu CRL mohla potom byť príliš veľká a preto sa objavili požiadavky na škálovateľnosť riešenia. Vznikli tzv. rozdielové CRL, ktoré obsahujú len certifikáty zneplatnené od doby vydania posledného CRL a sú teda menšie. Problém nastal aj s aktualizáciou certifikátov a preto vznikli rozšírenia CRL a certtifikátov informujúce o distribučných miestach CRL, adminstratívne práce spojené s CRL si vyžiadali možnosť delegovať právomoc vydávať CRL na inú certifikačnú autority a vznik tzv. nepriamych CRL. Iné rozšírenie CRL umožňuje vydávať tzv. obmedzené CRL, kedy sa CRL vzťahuje len na určitú podmnožinu certifikátov. Tieto rozšírenia boli realizované definovaním CRL rozšírení (podobne ako rozšírenia certifikátov verejného kľúča). Software pracujúci z CRL tieto rozšírenia s rôznou úspešnosťou implemetuje, štandard popisujúci CRL je vyžaduje len implementáciu rozšírenia Číslo CRL, ktoré udáva číslo vydávaného CRL.

Špeciálnym typom CRL sú CRL, ktoré obsahujú len certifikáty, ktoré boli zneplatnené od posledného vydania CRL. Takéto CRL sa volá rozdielové CRL. Vzniklo ako reakcia na problém s veľkosťou štandardných CRL. Rozdielové CRL popisujú len rozdiel oproti poslednému vydanému CRL, sú preto omnoho menšie. Pre úplnú informáciu o neplatných certifikátoch je však nutné získať aj CRL, voči ktorému bolo rozdielové CRL vydané. V praxi (Česká Republika) nie sú rozdielové CRL veľmi rozšírené.

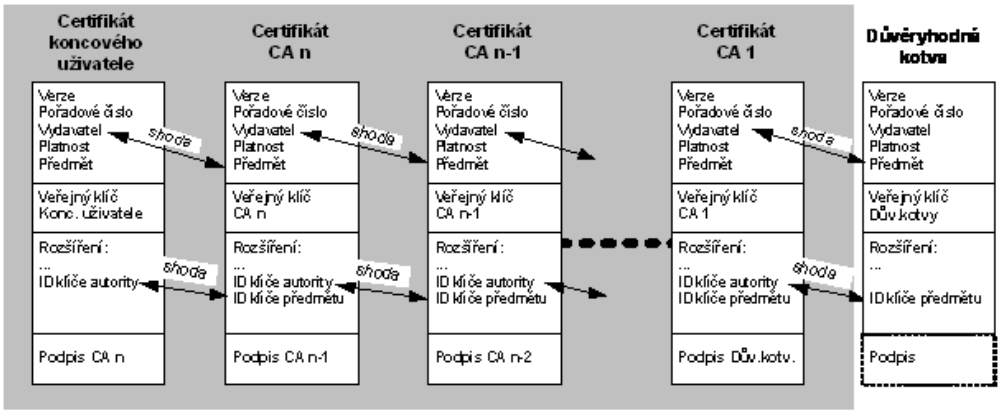
CRL je vydávané dávkovo a v praxi sú aktualizované niekoľko krát denne. Takáto perióda však nemusí každému vyhovovať, ak potrebuje poznať aktuálny stav certifikátu. Protokol OCSP umožňuje on-line kontaktovanie certifikačnej autority a umožňuje klásť dotazy na platnosť certifikátov. OCSP protokol je popísaný v štandarde RFC 2560. Podľa protokolu OCSP pošle klient certifikačnej autorite dotaz, ktorý obsahuje identifikáciu certifikátu. OCSP server certifikačnej autority odpovie, či je certfiikát platný, neplatný resp. táto informácia nie je dispozícií. Certifikačná autorita podobne ako v prípade CRL môže delegovať OCSP na iný server a vydá certifikát s príslušným rozšírením na podpisovanie OCSP odpovedí. Každá odpoveď musí byť OCSP serverom podpísaná. Toto si vyžaduje veľké požiadavky na výkon OCSP servera a škálovateľnosť tohoto riešenia z pohľadu počtu vydávaných OCSP odpovedí je diskutabilná. Problémy použitia OCSP protokolu sa vznikajú aj pri overovaní certifikátu. Ak je certifikát odvolaný, ale ešte nie je uverejnený na CRL, overovanie podľa OCPS a CRL môže dať rozdielne výsledky. Tento problém sa nedá riešiť obecne a rieši ho konkrétna certifikačná politika konkrétní certifikační autority. Štandard CAdES popisuje elektronické podpisy, ktoré môžu byť archivované dlhú dobu. Pre uľahčenie vyhľadávania CRL a OCSP potrebných pre určenie platnosti podpisovacieho certfikátu v dobe podpisu, umožňuje pripojiť k podpisu referencie resp. plné validačné dáta (CRL, OCSP odpovede).

2.6 Certifikačná cesta, koreňové certifikáty a overovanie platnosti certifikátu

Pre popísanie procesu overenie certifikátu je nutné zaviesť niekoľko pojmov:

Certifikát vydavateľa - jedná sa o certifikát, ktorého položka Predmet (Subject) je zhodná s položkou Vydavateľ (Issuer) overovaného certifikátu. Ak sú prítomné rozšírenia Identifikácia kľúča úradu na vydanom certifikáte a Identifikátor kľúča predmetu na certifikáte vydávateľa, musí byť ich hodnota rovnaká.

Dôveryhodná kotva - certifikát, ktorého dôveryhodnosť sa dá preukázať inou cestou. Dôveru je možné preukázať napríklad umiesnením certifikátu do skladu dôveryhodných certifikátov. Existuje rozdiel medzi platným a dôveryhodným certifikátom - platný certifikát je certifikát, ktorého plat-



Obrázok 2.1: Obrázok znázorňuje certifikačnú cestu certifikátu koncového užívateľa

nosť neskončila a nie je odvolaný. Dôveryhodný certifikát je taký, u ktorého je možné dôverovat údajom uvedeným v certifikáte o vlastníkovi a predmete certifikátu.

Certifikačná cesta - reťaz certifikátov, začínajúca užívateľským certifikátom a končiaca dôveryhodnou kotvou, kde po certifikáte nasleduje na ceste certifikát vydavateľa

Koreňový certifikát - jedná sa o certfikát, kde je vydavateľ certifikátu zhodný s predmentom certifikátu. Certifikát je teda podpísaný privátnym kľúčom, ktorého zodpovedajúci verejný kľúč sa certifikuje. Podpis má v tomto prípade zmysel len pre kontrolu integrity certifikátu a ako dôlaz o držaní zodpovedajúceho súkromného kľúča. Dôveryhodnosť takéhoto certifikátu sa dá vyjadriť len explicitne, nedá sa mu dôverovať na základe iných certifikátov. Dôveryhodné koreňové certikáty sú často dôveryhodnými kotvami.

Platnosť certifikátu - certifikát je platný, ak aktuálny čas neprekročil dobu platnosti certifikátu a certifikát nie je odvolaný. Informáciu o odvolaní certikátu je možné získať zo zodpovedajúceho CRL resp. OCSP odpovede.

Dôveryhodnosť certifikátu - certifikát je dôveryhodný, ak je certifikát vydavateľa dôveryhodný (tj. vydala ho dôveryhodná certifikačná autorita) a platný.

Z definícií sa dá odvodiť, že certikát je dôveryhodný a platný, ak sú všetky certifikáty certifikačnej cesty platné a koreň certifikačnej cesty tvorí dôveryhodná kotva.

Tvrdenie sa dá ukázať indukciu. Ak je overovaný certifikát platný a certifikát jeho vydavateľa je dôveryhodný a platný, je dôveryhodný a platný aj overovaný certifikát (to vychádza priamo z definícií). Pretože je dôveryhodná kotva dôveryhodná a platná (z predpokladov), môžeme na základe indukcie tvrdiť, že aj koncový certifikát je dôveryhodný a platný.

Z dôkazu je zrejmé, ako sa bude pri overovaní certifikátu postupovať. Štandard definuje, že pri overovaní certifikátu sa naskôr zostaví certifikáčná cesta ukončená dôveryhodnou kotvou a následne sa overí platnosť všetkých certifikátov (overenie platnosti začína u dôveryhodnej kotvy a postupuje smerom ku koncovému certifikátu). Ak sú všetky certifikáty platné, overenie koncového certifikátu prebehlo úspešne a certifikát je dôveryhodný a platný. Tento algoritmus nie je z implementačného hľadiska najšťastnejší. V praxi je pri tvorení certifikačnej cesty totiž zároveň možné overovať platnosť certifikátov, a tak je možné dôjsť ku prípadnej negatívnej odpovedi skôr. Štandard umožňuje použitie iných ako definovaného algoritmu na overenie certifikátu, ich záver však musí byť rovnaký. Vylepšený algoritmus overovania, ktorý je popísaný nižšie, zároveň konštruuje certifikáčnú cestu a overuje jednotlivé certifikáty:

1. koncový certifikát sa zaradí do certifikačnej cesty a overíme jeho platnosť

2. ak je posledný certifikát certifikačnej cesty dôveryhodna kotva, overenie prebehlo úspešne

3. vyhľadá sa certifikát vydavateľa certifikátu na konci certifikačnej cesty, overí sa jeho plastnosť a zaradí sa na koniec certifikačnej cesty

4. pokračuje sa bodom 2

Tento algoritmus zároveň konštruuje certifikačnú cestu až po dôveryhodnú kotvu a overuje platnosť jednotlivých certifikátov. Je ekvivalentný algoritmu overenie certifikátu, ktorý definuje štandard. Vykonáva rovnaké kroky v inom poradí, jednotlivé kroky sú však izolované a nemenia samotné dáta.

Pri overovaní certifikátu je dôležite sí uvedomiť, čo všetko určuje platnosť certifikátu:

- správne použitie súkromného kľúča. Certifikát zahŕňa informácie o povolených spôsoboch použita súkromného kľúča. Tieto informácie sú v rozšíreniach Použitie kľúča a Rozšírené použitie kľúča.

- rozšírenia certifikátu označené ako závažné. Ak nedokáže algoritmus spracovať rozšírenia označené ako závažné, nemôže byť použitý na overenie príslušného certifikátu.

- časová platnosť certifikátu - aktuálny čas je v intervale určenom časmi Platný od, Platný do, ktoré sú obsiahnuté v certifikáte
- certifikát nie je odvolaný. Certifikát nie je uvedený na CRL, prípadne OCSP odpoveď označila certifikát ako platný.

Ak zlyhá niektorý krok overovania platnosti certifikátu, certifikát je označený za neplatný a algoritmus overovania končí.

Proces overovania, či nie je certifikát odvolaný, musí brať do úvahy rôzne typy CRL, ktoré sú definované v štandarde (nepriame CRL, rozdielové CRL, obmedzené CRL). Táto práca však nepracuje so všetkými druhmi CRL a je algoritmus overenie certifikátu voči CRL jednoduchší.

Overenie, či nie je certifikát odvolaný je potom vykonané takto:

1. získa sa CRL, ktoré je platné (nenastal čas uvedený v poli nextUpdate)
2. overí sa, či certifikát zodpovedá CRL. Vydavateľ certifikátu musí byť zhodný s vydavateľom CRL
3. vytvorí a overí sa certifikačná cesta pre certifikát, ktorým bolo podpísané CRL
4. overí sa digitálny podpis CRL, teda či certifikát prislúcha CRL
5. prehliadneme sa CRL, či sa nenájde v zozname odvolaných certifikátov overovaný certifikát. Ak sa v zozname nenachádza, považuje sa za neodvolaný.

A popisu algoritmu je vidieť, že overovanie CRL zahrňuje overovanie certifikačnej cesty CRL. V praxi je však zriedkavé, aby bol certifikát overujúci CRL iný ako certifikátu vydavateľa overovaného certifikátu. Preto overovanie CRL a certifikátu prebieha ďalej overovaním vydavateľa certifikátu. Overením certifikátu vydavateľa zároveň overíme aj certifikát CRL.

Overovanie certifikátu k dátumu v minulosti

Prechádzajúca časť popisovala overovanie certifikátu. Overovanie k aktuálnemu času nemusí riešiť mnoho problémov, ktoré nastanú, ak je daná požiadavka na overenie certifikátu k nejakému dátumu v minulosti.

Ak je k aktuálnemu času certifikát platný, určite bol platný aj v minulosti (až na mechanismus pozasavení certifikátu). Ak je však certifikát v súčasnosti neplatný, nedá sa rozhodnúť o jeho platnosti a o tom, či bol privátny kľúč kompromitovaný, k nejakému času v minulosti.

Dôkaz platnosti certifikátu v minulosti sa dá zabezpečiť vydaním časového razítka, ktoré dokazuje existenciu certifikátu v dobe jeho platnosti. Dá

sa zabrániť aj zneužitiu súkromného kľúča: Zneužitý privátny kľúč sa použije a útočník oznámi, že ho použil v čase platnosti privátenho kľúča. Ako dôkaz doby použita kľúča je opäť možné požadovať platné časové razítko z dokumentu, pri tvorbe ktorého bol súkromný kľúč použitý.

2.7 Digitálny podpis

Digitálny podpis slúži na zabezpečenie integrity a autenticity správ. Pôvodne bol definovaný štandardom PKCS#7, neskôr bol prebratý ako internetový štandard CMS (Cryptgraphic Message Syntax) pod označením RFC-2315. Aktuálnym štandardom CMS je RFC-3852 [1]. Štandard CMS sa zaoberá zabezpečením správ obecne, nepopisuje len vytvorenie digitálneho podpisu, ale aj digitálnej obálky a pod. CMS správa obsahuje pôvodnú správu a jej typ, pridáva k nej ďalšie informácie (podpis, hash..), ktoré zabezpečia integritu alebo autenticitu pôvodnej správy. CMS správa sa môže stať súčasťou inej CMS správy.

Štandard CAdES pre dlhodobý elektronický podpis používa typ CMS správy pre digitálny podpis,.Štruktúra digitáneho podpisu pozostáva z týchto častí:

- verzia - udáva verziu digitálne podpísanej správy. Aktuálna verzia je verzia 4
- identifikátory hashovacích algoritmov použité pri tvorbe podpisu
- typ a obsah správy, ktorá má byť podpísaná. Samotná správa nemusí byť súčasťou štruktúry.
- nepovinne obsahuje zoznam certifikátov verejného kľúča, prípadne atribútových certifikátov. Mala by obsahovať certifikáty potrebné na overenie digitálneho podpisu, alebo aspoň podpisový certifikát.
- nepovinne obsahuje zoznam CRL. Zoznam by mal obsahovať všetky CRL potrebné na overenie certifikátov.
- zoznam podpisov - štuktúra SignerInfo

Podpísaná správa môže, ale aj nemusí, byť zahrnutá v podpise. Keď štruktúra neobsahuje podpisovanú správu, jedná sa o externý podpis. V prípade externého podpisu je pôvodná správa skladovaná oddelene, jej umiestnenie však musí byť známe.

Vlastný podpis popisuje štruktúrou SignerInfo. Digitálny podpis môže obsahovať viacero podpisov, čo zodpovedá prípadu viacerých podpisov na papierovom dokumente. Jednotlivé štruktúry SignerInfo pozostávajú z týchto položiek:

- identifikácia podpisového certifikátu - certifikát je identifikovaný na základe vydavateľa a poradového čisla certifikátu
- podpisované atribúty - tvorí ich množina atribútov, ktoré sú digitálne podpisané. Podpisované atribúty umožňujú do digitálneho podpisu zahrnúť aj ďalšie údaje (čas podpisu, idenfitikáciu certifikátu), ktoré sami nie sú súčasťou správy.
- algoritmus podpisu - definuje asymetrický algoritmus použitý k digitálnemu podpisu
- algoritmus otlačku - definuje použitý algoritmus pre výpočet otlačku správy resp. podpisovaných atribútov
- podpis - táto položka obsahuje vlastný digitálny podpis, tj. asymetrickou šifrou šifrovaný otlačok.
- nepodpisované atribúty - množina atribútov, ktoré nie sú digitálne podpísané

Integrita a autenticita podpisu sa zaisťuje zašifrovaním hash otlačku z dôležitých častí podpisu. Rozlišujú dva prípady:

- správa neobsahuje žiaden podpisovaný atribút - v tomto prípade sa podpis spočíta zašifrovaním otlačku dát zo správy
- správa obsahuje podpisované atribúty - v tomto prípade sa zašifruje otlačok spočítaný z podpisovaný atribútov. Podpisované atribúty však musia obsahovať atribút Otlačok správy (message digest), ktorý obsahuje otlačok pôvodnej správy, a atribút Typ obsahu (conent type). Otlačok podpisovanej správy je teda nepriamo zahrnutý v otlačku podpisovaných atribútov.

Podpisované a nepodpisované atribúty umožňujú do podpisu pridať rôzne typy atribútov. Ak je atribút pridaný medzi podpisované atribúty, je zaručená jeho integrita a autenticita (pomocou podpisu). Podpisovaný atribút sa však musí pridať pri vytváraní podpisu. Informácie, ktoré sú v čase podpisu neznáme, teda nie je možné pridať medzi podpisované atribúty. Nepodpisované atribúty môžu byť k podpisu pridávané neskôr, nie je však zaručená ich autenticita ani integrita.

Štandard CMS definuje niekoľko atribútov, mnoho atribútov však bolo defnovaných inými štandardmi. Príklady atribútov sú definované v tabuľke 2.1

Overenie digitálneho podpisu

Overenie digitálneho podpisu tu je chápané v zmysle overenie formálnej správnosti v rámci infraštruktúry PKI. Digitálny podpis môže obsahovať

Atribút	štandard	vnútorný/ externý podpis	Podpisovaný atribut
contentHints	ESS	oba	môže byť
contentReference	ESS	oba	musí byť
contentType	CMS	oba	musí byť
counterSignature	CMS	oba	nesmie byť
equivalentLabel	ESS	oba	musí byť
eSSSecurityLabel	ESS	oba	musí byť
messageDigest	CMS	oba	musí byť
msgSigDigest	ESS	iba vnúrorný	musí byť
mlExpansionHistory	ESS	iba vnúrorný	musí byť
receiptRequest	ESS	iba vnúrorný	musí byť
signingCertificate	ESS	oba	musí byť
signingTime	CMS	oba	musí byť
smimeCapabilities	S/MIME	oba	musí byť
sMIMEEncryption-KeyPreference	S/MIME	oba	musí byť

Tabuľka 2.1: Tabuľka uvádza ďalšie typy atribútov elektronického podpisu

viacero podpisov, ktoré sa overujú na sebe nezávisle. Overenie jedného podpisu potom vyzerá nasledovne:

1. overenie certifikátu podpisovanej osoby - podpisového certifikátu. Znamena to vytvorenie ceritikačnej cesty až ku dôveryhodnej kotve a overenie platnosti a dôveryhodnosti certifikátov.

2. verifikácia podpisu. Znamená to overiť štruktúru podpisu a overiť integritu podpisu:

   (a) spočítanie otlačku správy a jeho porovnanie s otlačkom uvedeným v podpisovaných atribútoch

   (b) dešifrovanie podpisu pomocou verejného kľúča uvedeného v podpisovacom certifikáte

   (c) porovnanie dešifrovaného podpisu s otlačkom z podpisovaných atribútov

Ak prebehli všetky body overenie bez chýb a otlačok z podpisovaných atribútov sa zhoduje s dešifrovaným podpisom, podpis je platný. Štandard popisuje takéto mechanické overenie, nič však nevraví o špecifických prípadoch, tieto špecifické prípady potom rieši politika elektronického podpisu.

Útoky na digitálny podpis

Táto práca sa zaoberá digitálnym podpisom a jeho archivovaním. V prípade, keď je digitálny podpis platný dlhú dobu, sa útočníkovy poskytuje dostatočný čas na napadnutie slabín tohoto podpisu. Je preto dôležité tieto slabiny popísať.

Orezanie podpisov - digitálny podpis môže obsahovať viac podpisov. Tieto podpisy sú na sebe nezávislé a žiadne mechanizmus nezaručuje ich integritu. Útočník môže podpisy do správy pridávať, ale aj odoberať, bez toho, aby za sebou zanechal stopu. Digitálny podpis by si preto mali jednotlivé strany archivovať na bezpečnom mieste.

Zmena nepodpisovaných častí digitálneho podpisu - podpis zaručuje len integritu častí podpisu, z ktorých bol spočítaný otlačok. Takto je možné zaručiť len integritu vlastnej správy a podpisovaných atribútov. Ostatné položky digitálneho podpisu môžu byť útočníkom zamenené. Týka sa to položiek ako identifikácia podpisovacieho certifikátu, zoznamu certifikátov a CRL. Ochranou proti tomu útoku je vloženie dôležitých informácií do podpisovaých atribútov. Niekedy však toto nie je možné, nakoľko nie sú pri tvorení podpisu tieto informácie známe.

Čas podpisu - čas podpisuje môže byť pridaný podpisovanou osobou do podpisovaných atribútov. Podpisovaná osoba však môže do podpisu vložiť ľubovolný čas. Takto môže vydávať dokument za podpísaný v čase, keď ho ešte nepodpísala. Ochranu proti takejto forme zneužitia tvorí časové razítko, ktoré spája existenciu podpisu s časom z dôveryhodného zdroja.

2.8 Časové razítko

Dôveryhodné časové razítko je vydané dôveryhodnou časovou autoritou a dokazuje existenciu dát v určitým čase. Tento čas je udaný v časovom razítku. Štandard popisujúci časové razítka je RFC 3161 [11] Postup vytvorenia časového razítka je nasledovný:

1. užívateľ vytvorí žiadost o časové razítko. Do žiadost zahrnie hash otlačok dokumentu, pre ktorý chce razítko získať

2. časová autorita po obdržaní žiadosti od užívateľa vytvori štuktúru, tzv. TimeStampInfo. Do tejto štruktúry zahrnie hash dokumentu, aktuálny čas z dôveryhodného zdroju času uvedený v žiadosti, politiku vydávania časových razítok a dodatočné informácie

3. časová autorita štruktúru TimeStampInfo digitálne podpíše, čím vytvorí časové razítko. Tým zabezpečí autentickosť a integritu štruktúry.

Časové razítko, ktoré má formát CMS správy, je zaslané späť užívateľovi v štruktúre nazvanej Timestamp Response. V prípade, že sa nepodarí vytvoriť časové razítko, obsahuje štruktúra Timestamp Response dôvod chyb, ktorá pri vytváraní nastala.

Pri overení časového razítka sa najprv overí digitálny podpis na základe certifikátu časovej autority. Potom sa porovná hash otlačok zo štruktúry TimeStampInfo z hashom dokumentu. Ak sú tieto otlačky zhodné, dokument existovať pred časom uvedeným v štuktúre TimeStampInfo. Platnosť časového razítka je teda určená platnosťou certifikátu časovej autority, ktorá razítko vydala.

Certifikát časovej autority má oproti bežným certikátom niekoľko špecifík. Certifikát musí obsahovať v rozšírení Použitie kľúča, že sa jedná o certifikát časovej autority. Tento certifikát ma tiež väčšinou dlhšiu dobu platnosti ako bežné vydávané certifikáty pre užívateľov. Používa sa totiž na predlžovanie platnosti digitálneho podpisu, a teda by mal mať dlhšiu platnosť ako podpisovací certifikát. Časové razítka majú mnoho spôsobom využitia (razítkovanie auditných záznamov, razítkovanie dokumentov), využívajú sa však najmä na predlžovanie platnosti digitálneho podpisu.

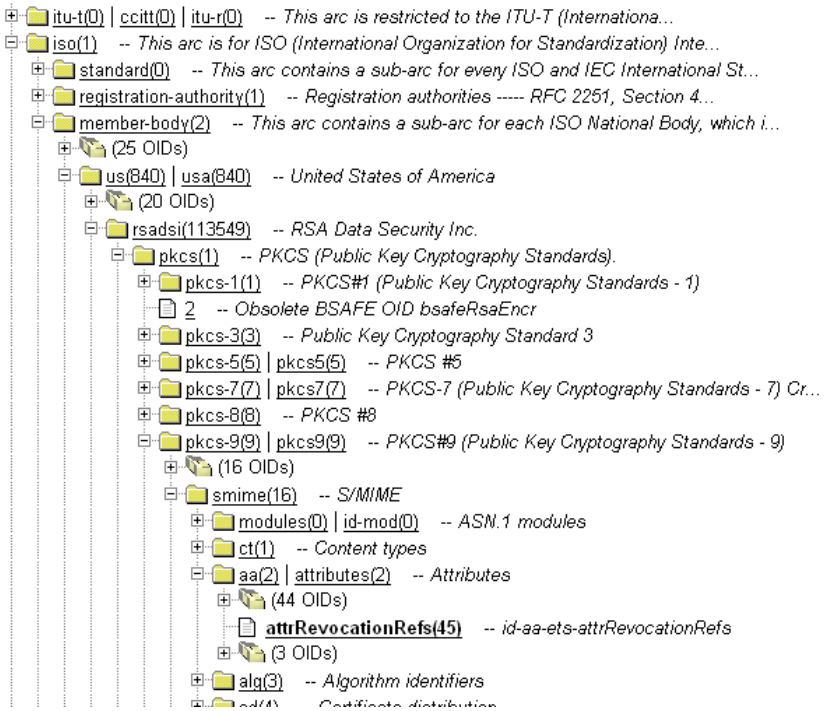
Štandard CAdES popisuje predlžovanie podpisu pridávaním časových razítok k podpisu. Tieto časové razítka sú teda tiež dlhodobo archivované spolu s podpisom. Štandard CAdES preto popisuje atribúty, ktoré je možné pridať k časovému razítku. Tieto atribúty obsahujú validačné referencie a aj úplné dokumenty potrebé na overenie razítka (certifikáty, CRL) a zjednodušujú ich hľadanie v budúcnosti.

2.9 Problémy spojené platnosťou digitálneho podpisu

Digitálny podpis je neplatný po uplynutí doby platnosti podpisovacieho certifikátu. Ak je štruktúra digitálneho podpisu orazítkovaná v dobe platnosti digitálneho podpisu, jej existencia je v dobe platnosti dokázateľná. Ak je teda časové razítko podpisu platné, podpis je možné vyhlásiť tiež za platný, a kedže certikáty časových autorít majú dlhšiu platnosť ako bežné certifikáty, predĺženie platnosti digitálneho podpisu môže byť významné.

Pri prípadnej požiadavke na predĺženie platnosyi podpisu aj po uplynutí plastnosti časového razítka, je možné vydať nové časové razítko. Nakoľko je časové razítko forma elektronického podpisu, orazítkovaním časového razítka predĺžime jeho platnosť. Pri prípadnom riziku prelomenia hashovacích algoritmov použitých pri tvorbe digitálneho podpisu, môžeme do výpočtu hashu pre časové razítko zahrnúť aj pôvodnú správu.

Takáto úvaha je základom pre predlžovanie platnosti digitálneho podpisu podľa štandardu CAdES, ktorý rieši implementačné detaily tohoto mechanizmu.



Obrázok 2.2: Obrázok znázorňuje strom registrovaných objektov

2.10 Identifikácia objektov

Mnohé štandardy popisujú objekty, ktoré musia byť jednoznačne definované. Jednoznačnosť identifikácie objektov sa dosahuje ich registrovaním u špeciálnych normalizačných autorít. Autority, ktoré umožňujú registráciu objektov sú ITU a ISO, ktoré jednoznačne spoja objekt s jeho identifikátorom, tzv. OID.

Objekty sú klasifikované v stromovej štruktúre, na vrchole ktorej sú samotné normalizačné úrady. Zaradenie objektu do podstromu príslušného úradu vyjadruje, kto tento objekt registroval. Uzlom stromu sú pridelené číselné hodnoty a čísla na ceste od koreňa stromu po príslušný objekt vyjadruje identifikáciu tohto objektu (OID). Identifikácia objektov má veľký význam, používa sa nielen v štandardoch PKI, ale aj v rôznych adresárových službách (X500, LDAP) alebo dokonca v zdravotníctve. Štandard PKI používa identifikáciu objektov na identifikovanie hodnôt (v položke Vydavateľ), identifikáciu hashovacích algoritmov ale aj na určenie typu podpisovaných a nepodpisovaných atribútov v štrutúre CMS.

2.11 ASN1, BER a Base64

Digitálne dokumenty, ktoré sú súčasťou infraštruktúry PKI sú štruktúry a na popísanie týchto štruktúr sa používa jazyk ASN1 (Abstract Syntax No-

tation One). Jazyk ASN1 je štandardizovaný a flexibilný jazyk popisujúci datové štruktúry. Popisuje objekty spôsobom nezávislým od počitačových platforiem. Tento jazyk definuje jednoduché typy a spôsob, akým sa z jednoduchých typov vytvárajú typu zložené. Jazyk ASN1 však nedefinuje spôsob kódovania štruktúr do iných formátov (binárne, XML).

Do binárnej formy sú štruktúry kódované pomocou jazyka DER (Distinguished Encoding Rules) alebo BER. Kódovanie DER vzniklo ako zjednodušenie kódovanie BER, tvorí teda jeho podmnožinu. Štadardy PKI používajú práve kódovanie DER pre zápis ASN1 štruktúr.

Výhodou kódovania DER je úsporný spôsob zápisu, dáta sa však nedajú prenášať sedem bitovým kanálom. Kódovanie Base64 naproti tomu umožňuje zasielať dáta e-mailom, štruktúra v kódovaní Base64 je však asi o tretinu dlhšia ako v kódovaní DER. Kódovanie Base64 bolo vydané v štandarde PEM, preto sa tak aj teraz tento typ kódovania označuje ako PEM formát.

Kapitola 3

Analýza formátov podpisov podľa štandardu CAdES

Štandard CAdES (tj. ETSI TS 101 733) popisuje niekoľko typov podpisov. V nasledujúcej časti budú popísané jednolivé typy podpisov, ich význam a použitie.

Štandard dlhodobého elektronického podpisu len rozširuje existujúci štardard CMS [1], pridáva nové podpisované a nepodpisované atribúty.

Štandard CAdES definuje dva základné elementární druhy podpisu:

- Basic Electronic Signature - CAdES-BES
- Explicit policy-based Electronic Signature - CAdES-EPES

Ďalšie typy podpisov naväzujú na typ BES resp. EPES. Podpisy typu BES resp. EPES zavádzajú podpisované atribúty a preto ich musí vytvoriť už podpisovateľ.

3.1 CAdES-BES

Podpis typu BES je základný typ podpisu, ktorý definuje štandard CAdES, a musí obsahovať:

- podpisované dáta[1])
- súbor povinných podpisovaných atribútov, ktoré definuje štandardy CMS a ESS
- súbor pridaných podpisovaných atribútov, ktoré definuje štandard CAdES
- hodnotu digitálneho podpisu spočítanú podľa štandardu CMS

Povinné podpisované atribúty sú:

[1]Štandard CMS povoľuje aj podpis bez dát

- Content-type - je definovaný v RFC 3852 a špecifikuje typ správy
- Message-digest - je definovaný v RFC 3852 a obsahuje hash otlačok podpísanej správy
- ESS signing-certificate alebo other-signing-certificate - tieto atribúty sú definované v štandarde ESS resp. CAdES a obsahujú identifikátor certifikátu a jeho hash otlačok. Hash otlačok certifikátu v ESS signing-certificate je tvorený pomocou algoritmu SHA-1, v prípade other-signing-certificate je algoritmus tvorby hash otlačku voliteľný.

V podpise tvorenom podľa štandardu CMS je v položke Idenfikácia podpisu (v štruktúre SingerInfo) identifikovaný podpisový certifikát. Na jej základe je možné jednoznačne identifikovať príslušný certifikát a použiť ho pri overovaní podpisu. Identifikácia ceritifkátu však nie je podpísaná a preto ju môže útočník kompromitovať a nahradiť. Štandard CAdES preto zavádza povinné podpisované atribút ESS signing-certificate resp. other-signing-certificate, aby nemohla byť infomácia o podpisovacom certifikáte podvrhnutá. Podpis typu BES musí obsahovať aspoň jeden z týchto atribútov. Podpisový certifikát je v atribúte `ESS Signing Certifikcate` resp. `Other Signing Certifikcate` identifikovaný hash otlačkom z certifikátu, prípadne identifikáciou vydavateľa certifikátu a sériového čísla.

Štandard CAdES definuje aj ďalšie podpisované atribúty, niektoré z nich sú popísané v tabuľke 3.1.

Atribut	**Popis**
Signer-location	Udáva miesto vzniku podpisu
Signer-attribute	Umožňuje vložiť atribútový certifikát, ktorý určuje oprávnenia podpisovateľa
Content-time-stamp	Časové razítko z podpísanej správy. Razítko dokazuje existenciu správy v čase.
Commitment-type-indication	Udáva spôsob zainteresovanosti podpisujúceho (či správu len podpísal, alebo ju vytvoril atď.)

Tabuľka 3.1: Tabuľka uvádza ďalšie podpisované atribúty definované štandardom CAdES

Podpis typu BES je základným typom podpisu definovaným v štandarde CAdES. Podpis BES neobsahuje dosť informácií, aby mohol byť overený v dlhšom časovom horizonte. Tento podpis však spĺňa všetky zákonné požiadavky určené Európskou smernicou pre elektronické podpisy a zaručuje základnú ochranu autenticity a integrity.

Obrázok 3.1: Ilustrace podpisu typu CAdES-BES

3.2 CAdES-EPES

Keďže implementácia podpisu typu EPES nie je predmetom tejto diplomovej práce, bude zmienený len stručne. Certifikačné autority postupujú pri vydávání certifikátov podľa certifikačnej politiky a v certifikáte je možné identifikovať certifikačnú politiku v rozšírení Certificate Policies. Štandard CAdES zavádza atribút, ktorý umožní definovať aj politiku elektronického podpisu - definuje rozšírenie elektronického podpisu o podpisovaný atribút `signature-policy-identifier`. Tento atribút obsahuje identifikátor politiky a hash dokumentu (hashovací algoritmus je voliteľný) podpisovacej politiky a tým ju jednoznačne identifikuje. Pridaním tohoto atribútu podpisovateľ vyjadruje, že pri tvorbe podpisu aplikoval konkrétnu politiku. Zaradenie atribútu medzi podpisované atribúty zabraňuje prípadnému neskoršiemu podvrhu.

3.3 Elektronické podpisy obsahujúce validačné dáta

Podpis typu BES zaisťuje základnú ochranu integrity a autenticity, neobsahuje však žiadne validačné dokumenty (až na podpisový certifikát) potrebné pre neskoršie overenie podpisu. Validačné údaje tvoria:

- certifikáty
- informácie o zneplatnení certifikátov - CRL
- časové razítko podpisu dokladujúce dobu vzniku podpisu (resp. časová značka v auditnom logu časovej autority)
- prípadne politiku digitálneho podpisu, ak bola použitá

Motiváciou pre zahrnutie takýchto informácií do podpisu bol problém, ktorý sa objavoval pri pozdejšom overovaní podpisov. Štandard CMS nepodporuje uloženie validačných informácií a pri dlhodobej archivácií podpisu je ich uloženie spolu s podpisom žiadúce. Môže totiž dôjsť k ukočeniu platnosti

certifikátov a tým sa ich dohľadanie podstatne sťaží. Preto bolo žiadúce, aby vznikli štruktúry, ktoré by boli súčasťou podpisu a uloženie validačných dát by umožnili.

Veľmi dôležité je uvedomiť si, aké validačné dáta su vlastne potrebné. K validácií podpisu je potrebný podpisovací certifikát, súbor certifikátov až po tzv. self-signed certifikát (tzv. certifikáčná cesta) a informácia o tom, či sú certifikáty v dobe podpisu platné (nie sú expirované a odvolané). Je tiež nutné rozlíšit v akej podobe budú validačné dáta uložené, či je potrebné pripojiť k podpisu úplný zoznam certifikátov, CRL a OCSP odpovedí, alebo postačuje pripojiť k podpisu len zoznam referencií na tieto dokumenty a samotné dokumenty archivovať samostatne.

S odvolávaním certfikátov sú spojené ďalšie problémy. CRL sú vydávané certifikačnými autoritami v širokých intervaloch (1-2x denne), čo je pre útočníka, ktorý získal súkromný kľúč obete, dostatočný čas, aby stihol napáchať škody. Aby bola platnosť certifikátu nespochybniteľná, treba brať do úvahy až CRL vydané po čase vzniku podpisu (resp. časového razítka dokladujúcom čas existencie podpisu). Doba od získania dôkazu o existencií podpisu (časové razítko, časová značka) po čas vydania relevantného CRL sa nazýva grace period. Až po tejto dobe sú validačné dáta relevantné a môžu byť pridané k podpisu.

Štandard CAdES definuje nepodpisované atribúty, ktoré umožnia do podpisov typu BES resp. EPES vložiť referencie alebo kompletné validačné dáta. Validačné údaje takto môže byť do podpisu zahrnuté podpisovateľom alebo schvaľovateľom, mali by však byť zahrnuté ešte pred archiváciou podpisu.

Podľa typu validačných dát, ktoré je možné k podpisu pridať rozlišujeme niekoľko typov podpisov:

- Electronic Signature with Time - CAdES-T Elektronický podpis obsahuje časové razítko dokladujúce čas vzniku podpisu
- ES with Complete validation data references - CAdES-C Elektronický podpis obsahuje referencie na certifikáty a CRL

Okrem validačných dát je do podpisu možné zahrnúť aj iné informácií, ktoré umožňujú jeho dlhodobú archiváciu a zachovanie platnosti aj po dlhej dobe. Jedná sa o tieto typy podpisov:

- Extendes Long Electronic Signature - CAdES-X Long Elektronický podpis obsahuje zoznam certifikátov a CRL (nielen referencie)
- Extended Electronic Signature with Time Type 1 - CAdES-X Type 1 Elektronický podpis C obsahujúci razítko celej štruktúry
- Extended Electronic Signature with Time Type 2 - CAdES-X Type 2 Elektronický podpis Long obsahujúci razítko z referencií
- Archival Electronic Signature - CAdES-A Elektronický podpis umožňuje prerazítkovávnie a predlžovanie platnosti

3.4 CAdES-T

Podpis tohoto typu obsahuje dôveryhodný čas spojený s podpisom. Tento čas môže byť zabezpečený:

- nepodpisovaným atribútom `signature-time-stamp` pridaným do elektronického podpisu
- časovou značkou (napr. záznam v auditných logoch časovej autority)

Atribút signature-time-stamp obsahuje časové razítko z hodnoty elektronického podpisu. V prípade časovej značky je časová autorita povinná dodať dôkaz o existencii podpisu v danom čase.[2] Tento typ podpisu teda jednoznačne potvrdzuje existenciu podpisu pred časom uvedeným v časovom razítku.

Štandard odporúča pridanie atribútu už podpisovateľom, prípade overovateľom hneď po obdržaní podpisu. Určite by však tento atribút mal byť pridaný pred expiráciou certifikátu prípadne jeho zneplatnením.

Podpis typu CAdES-T je možné overiť aj po skončení platnosti podpisovacieho certifikátu. Stačí len dokázať, že v dobe určenej časom v časovom razítku bol podpis platný (certifikát nevypršal, nebol odvolaný...) a časové razítko je súčasnosti platné. Dôkaz o existencii podpisu v dobe keď bol platný implikuje jeho súčasnú platnosť.

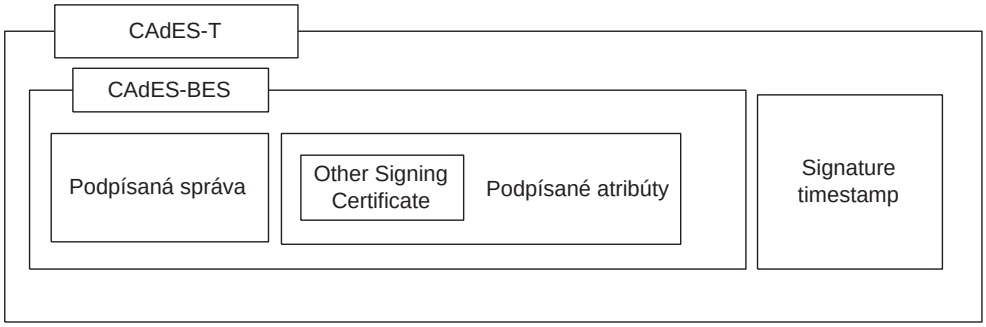
Pri použití časového razítka nie je doba platnosti podpisu určená dobou platnosti podpisovacieho certifikátu, ale platnosťou certfikátu časovej autority. Doba platností kvalifikovaných certifikátov u ICA je maximálne jeden rok, pri certifikátoch časovej autority je to v súčasnosti päť rokov, čo je významný rozdiel.

Pri množsve požiadavkov na archiváciu je doba 5 rokov dostatočná a podpis typu CAdES-T je postačujúci. Po dobe platnosti certifikátu časovej autority v časovom razítku je toto razítko a tým aj celý podpis neplatný. Pre ďalšie predlženie platnosti je nutné použiť typ podpisu CAdES-A, ktorý bude popísaný neskôr, podpis CAdES-T je teda vhodný skôr na krátkodobú archiváciu (do päť rokov).

3.5 CAdES-C

Elektronický popis podľa štandardu CMS nemusí obsahovať žiadne dodatočné údaje potrebné pre validáciu podpisu. Pri krátkodobej archivácií to ani nebýva problém - všetky potrebné informácie sú v dispozícií, naviac povinné pridávanie týchto informácií by pri bežnom styku CMS správy zbytočne zväčšovalo.

[2] K podpisu nie je pridaný žiaden atribút, význam časovej značky je však rovnaký ako atribútu `signature-time-stamp`.



Obrázok 3.2: Ilustrace podpisu typu CAdES-T

Pri požiadavke na dlhší čas archivácie sa však validačné dáta môžu hodiť. Po čase už totiž nebudú bežne v dispozícií a ich hľadanie v archívoch certifikačných autorít bude pracné. V taktomto prípade sa hodia aspoň referencie na potrebné validačné dáta a práve podpis typu CAdES-C tieto referencie zavádza. Vlastné validačné dáta (certifikáty, CRL a OCSP) potom môžu byť uložené samostatne, čo znižuje veľkosť uloženého elektronického podpisu. Referenciu na príslušný dokument (certifikát, CRL) tvorí identifikátor tohoto dokumentu a hash otlačok z tohoto dokumentu, príslušný dokument je teda jednoznačne identifikovaný.

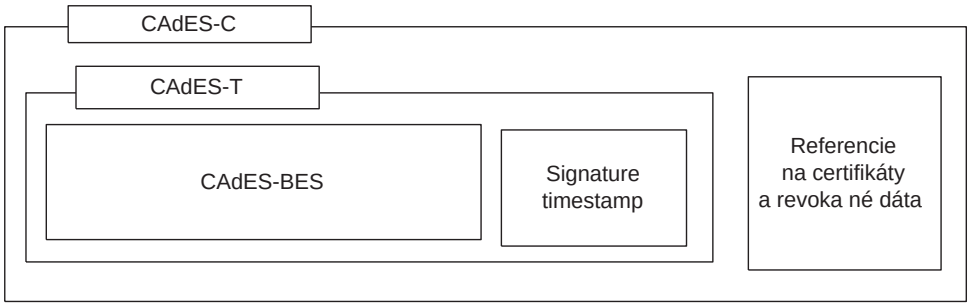
Podpis s komplenými referenciami na validačné dáta je rozšírením podpisu typu CAdES-T a pridáva tieto nepodpisované atribúty:

- `complete-certificate-references` - obsahuje referencie na všetky certifikáty v certifikačnej ceste použitej na overenie podpisu
- `complete-revocation-references` - obsahuje referencie na všetky CRL a OCSP odpovede použité pri overení podpisu

Jedná sa o nepodpisované atribúty a preto ich do podpisu môže pridať aj overovateľ. Mali by však byť pridané hneď, ako to je možné - teda v čase, keď budú v dispozícií validačné dáta po uplynutí tzv. grace period. Nakoľko sa jedná o nepodpisované atribúty existuje možnosť útoku nahradením.

Po expirácií certifikátu autority, ktorá vydala CRL a OCSP odpovede, môže byť zneužitý jej privátny kľúč. Útočník tak môže vytvoriť falošné validačné údaje a zmeniť príslušné referencie v podpise, čím podpis kompromituje. Tomuto zneužitiu sa dá zabrániť, ak sa aj na referencie validačných dát získa časové razítko, čo doloží ich existenciu v danom čase. Toto razíko pridáva typ podpisu CAdES-X Type 1, ktorý bude popísaný neskôr.

Podpis typu CAdES-C obsahuje len referencie na použité validačné dáta. Ak existuje riziko straty certifikátov prípadne CRL, zoznam referencií nie je dostatočný - je potrebné vložiť do podpisu zoznam certfikátov a CRL, čo umožňuje typ podpisu CAdES-X Long.



Obrázok 3.3: Ilustrace podpisu typu CAdES-C

3.6 CAdES-X Long

Tento typ podpisu je rozšírením typu CAdES-C, umožňuje však do podpisu zahrnúť aj kompletné validačné dáta (certifikáty, CRL a OCSP). Takto bude úložište dokumentov potrebných pre overenie podpisu súčasťou podpisu, čo zabraňuje strate týchto informácií.

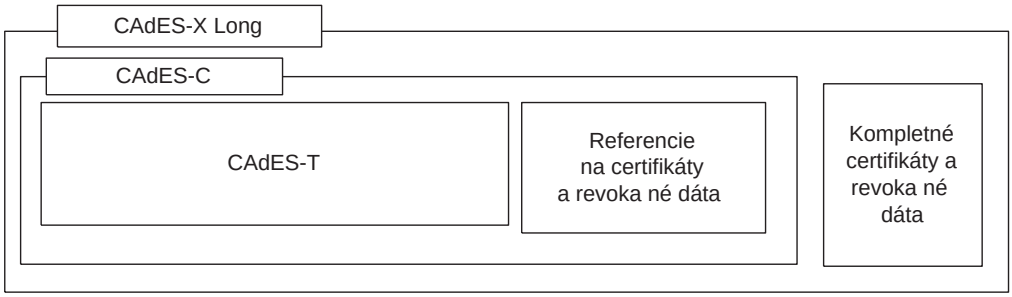
Podpis je rozšírením podpisu typu CAdES-C a pridáva tieto nepodpisované atribúty:

- `certificate-values` - obsahuje všetky certifikáty referencované v complete-certifikcate-references
- `revocation-values` - obsahuje CRL, OCSP odpovede a iné revokačné dokumenty referencované v atribúte complete-revocatiom-references

Tieto atribút by mal obsahovať certifikáty a CRL, ak hrozí riziko straty týchto údajov. Certifikáty a CRL uložené v podpise by sa však v žiadnom prípade nemali pokladať za dôveryhodné, aby bolo možné predchádzať nasledovnému typu útoku.

Útočník vytvorí vlastné certifikáty a CRL so zhodnými identifikačnými údajmi ako má dôveryhodná certifikačná autorita a obeť (vydavateľ, predmet...), súkromné a verejné kľúče nebudú so skutočnými kľúčmi certifikačnej autority a obete súhlasiť. Na základe týchto dokumentov podpíše kompromitovaný dokument a ďalej ho archivuje podľa štandardov CAdES. Po dlhej dobe vytvorí až podpis typu CAdES-A a v tomto momente podpis zverejní. Ak pri overovaní obeť nevlastní jej starý certifikát, resp. nie je možné zadovážiť pôvodný certifikát certifikačnej autority, nie je možné overovanie nijak napadnúť a overovanie skončí úspešne. Certifikáty v atribúte certificate-values by preto mali byť overené voči dôveryhodnému certifikátu, ktorý je získaný inou cestou.

Spôsob overenia podpisu sa však môže popisovať konkrétnou politikou digiálneho podpisu a ak atribút certificate-values vkladá dôveryhodná osoba, je možné mu dôverovať. Knižnica LTES preto bude umožnovať zvoliť, či zaradiť certifikáty z atributú certificate-values medzi dôveryhodné, alebo bude vyžadovať dôveryhodný certifikát z iného zdroja.



Obrázok 3.4: Obrázok ilustruje podpis typu CAdES-X Long

Poznámka: Časové razítka majú tiež formu elektronického podpisu, pre potreby dlhodobej archivácie môžu tiež obsahovať nepodpisované atribúty certificate-values a revocation-values.

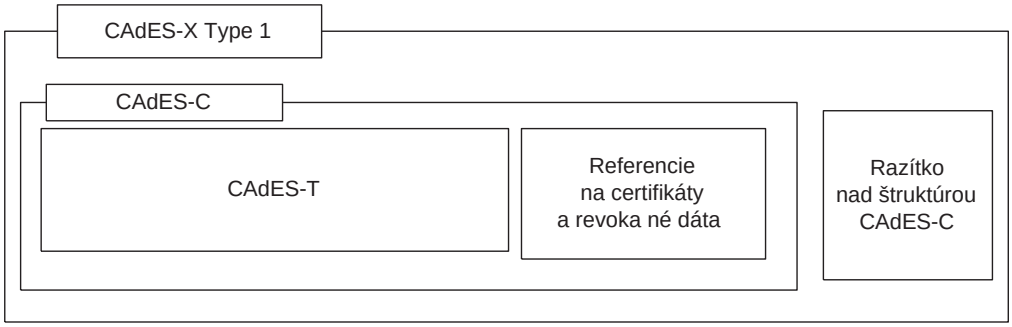
3.7 CAdES-X Type 1

Tento typ podpisu rozširuje podpis typu CAdES-C a pridáva nepodpisovaný atribút `CAdES-C-time-stamp` obahuje časové razítko celej štruktúry podpisu CAdES-C.

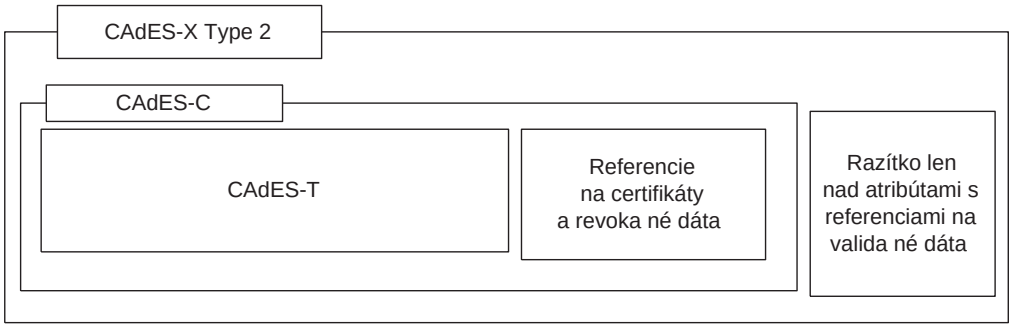
Pridanie časového razítka zabezpečí integritu a dôkaz existencie v čase celého podpisu včetne referencií na validačné dáta. Okrem iných častí podpisu dokazuje existenciu referencií na validačné dáta k času. To zabráni útočníkovi vložiť referencie na podvrhnuté certifikáty a CRL v dobe, keď bude súkromný kľúč certifikačnej autority kompromitovný. Časové razítko by preto malo byť pridané hneď po vytvorení referencií, po kompromitácií súkromného kľúča je už pridanie časového razítka zbytočné.

Automatizované nástroje na overenie elektronického podpisu nedokážu zistiť čas kompromitácie súkromného kľúča certifikačnej autority, mechanizmus CRL je totiž v tomto prípade nepoužiteľný, nakoľko aj samotné CRL už môžu byť podvrhnuté. Overovateľ by si mal preto vždy ručne skontrolovať čas vydania tohoto časového razítka a porovnať ho s časom prípadnej kompromitácie súkromného kľúča autority a tým overiť, či neboli referencie na validačné dáta pridané až po kompromitácií. Automatizované nástroje by mali čas vydania takéhoto časového razítka zobrazovať.

Nakoľko sa jedna o nepodpisovaný atribút, útočník ho môže z podpisu odstrániť. Ak je prijatý elektronický podpis bez časového razítka nad validačnými referenciami cez nedôveryhodný kanál a validačné údaje už nie je možné overiť z iného dôveryhodného zdroja (certifikačná autorita), existuje riziko, že sú sfalšované.



Obrázok 3.5: Ilustrace podpisu typu CAdES-X Type 1



Obrázok 3.6: Ilustrace podpisu typu CAdES-X Type 2

3.8 CAdES-X Type 2

Tento typ podpisu rozširuje podpis typu CAdES-C a pridáva nepodpisovaný atribút `CAdES-C-time-stamped-certs-crls-references` obsahuje časové razítko z referencií validačných dát (atribúty complete-certificate-references, complete-revocation-references).

Pridanie časového razítka zabezpečí integritu a dôkaz existencie referencií na validačné dáta. Tieto referencie teda nemôžu byť neskôr pri prípadnej kompromitácií privátneho kľúča certifikačnej autority podvrhnuté. Rovnako ako pri type podpisu CAdES-X Type 1 hrozí odstránenie časového razítka, nakoľko sa jedná o nepodpisovaný atribút. Aj keď Type 2 obsahuje len časové razítko z referencií, oba typy čelia rovnakým bezpečnostným hrozbám (viď CAdES- Type 1). V praxi je však použiteľnejší Type 2, pretože nie je nutné získavať razítko pre každý samostatný podpis, ale stačí získať razítko validačných referencií a toto je možné opakovane používať pri viacerých podpisoch. Príkladom môže byť firma, kde sa medzi dvoma vydaniami CRL validačné referencie nemenia, a tak môžu zamestnanci používať tieto referencie a prislúchajúce časové razítko viacnásobne.

3.9 CAdES-X Long Type 1 alebo Type 2

Tento typ podpisu je kombináciou podpisu typu CAdES-X Long a jedného z typov CAdES-X Type 1 resp. CAdES-X Type 2.

Kombinácia oboch typov prináša výhody použitia každého z nich - validačné referencie sú chránené časovým razítkom a zároveň nehrozí strata príslušných validačných dokumentov (certifikát, CRL), pretože sú súčasťou podpisu.

3.10 CAdES-A

Ak je elektronický podpis archivovaný dlhú dobu, časové razítka, ktoré zaručujú platnosť podpisu po expirácií podpisového certifikátu sa stanú časom neplatné, či už pre slabý hashovací algortimus alebo dobu platnosti razítka. Potom je nutné razítkovať elektronický podpis opakovane. Toto umožňuje archívna forma elektronického podpisu. Tento typ podpisu završuje štandard CAdES a poskytuje prakticky ľubovolne dlhú dobu archivácie.

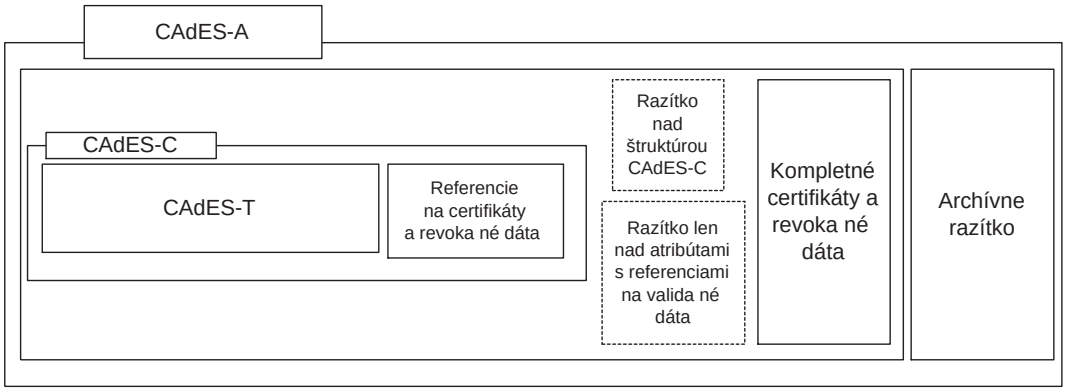
Archívna forma elektronického podpisu môže byť postavená na ktoromkoľvek z rozšírených podpisov typu CAdES-X Long (CAdES-X Long Type 1 resp. Type 2).

Tento typ zavádza nepodpisovaný atribút `archive-time-stamp`. Podpis môže obsahovať viacero takýchto atribútov a slúži k predĺženiu platnosti podpisu.

Atribút obsahuje časové razítko, ktoré je vydané na hash reťazca, ktorý vznikne spojením nasledovných:

- vlastnej podpisovanej správy obsiahnutej v štruktúre encapContentInfo, ktorá je súčasťou podpisu
- prvky Certificates a crls zo štruktúry podpisu
- spolu z celou štruktúrou SingerInfo, teda aj podpisovanými a nepodpisovanými atribútmi

Archívne časové razítko predĺží platnosť podpisu a tá bude zhodná z dobou platnosti pridaného archívneho časového razítka. Platné archívne časové razítko totiž dokazuje existenciu podpisu v dobe, keď bol ešte platný. Je nutné dokázať aj existenciu správy, ktorá je podpísaná, a preto je táto správa súčasťou reťazca, pre ktorý sa získava časové razítko. Zahrnúť pôvodnú správu do výpočtu časového razítka je nutné z dôvodu možného oslabenia algoritmu pre výpočet otlačkov. Existenciu pôvodnej správy totiž dokladá len jej otlačok v podpisovanom atribúte messageDigest, ktorý je súčasťou podpisu. Pri potencionálnom oslabení algoritmu pre výpočet otlačkov by mohol útočník vytvoriť správu s rovnakým otlačkom ako je ten, uvedený v messageDigest a takto by mohol pôvodnú správu podvrhnúť.



Obrázok 3.7: Ilustrace podpisu CAdES-Archive

3.11 Popis atribútov

V nasledovnej časti budú popísané atribúty, ktoré je nutné implementovať, aby bolo možné vytvárať všetky druhy podpisu, ktoré CAdES definuje. Štandard CAdES zavádza aj iné atribúty, tie však majú len pomocný charakter a nie sú pre implementáciu jednotlivých typov podpisov nutné.[3]

Atribút Other Signing Certificate

Jedná sa o podpisovaný atribút. Tento atribút jednoznačne identifikuje podpisový certifikát na základe hash otlačku certifikátu. Atribút umožňuje pre výpočet hash otlačku použiť ľubovolný hash algoritmus. Okrem identifikácie certifikátu na základe hash otlačku, môže atribút obsahovať aj vydavateľa a sériové čislo podpisovacieho certifikátu, táto identifikácia zjednodušuje identifikáciu certifikátu pre užívateľa. Hash zahrnutý v atribúte sa získa ako hash otlačok z DER kódovania podpisovacieho certifikátu.

Poznámka: Aj keď môže atribút obsahovať identifikáciu viacerých certifikátov, podľa štandardu CAdES by mal obsahovať len identifikáciu podpisového certifikátu.

Atribút ESS Signing Certificate

Jedná sa o podpisovaný atribút. Tento atribút, rovnako ako atribút Other Signing Certificate identifikuje podpisový certifikát na zákalade hash hodnoty. Pre výpočet otlačku z certifikátu však podporuje len SHA1 algoritmus, pri použití silnejších algoritmov je preto nutné použiť atribút `Other Signing Certificate`.

[3]Atribút ESS Signing Certificate je definovaný v štandarde ESS (RFC 5035) a preto nebude popísaný.

Poznámka: Aj keď môže atribút obsahovať identifikáciu viacerých certifikátov (ako napríklad pri identifikácií celej certifikačnej cesty v štandarde TSA), podľa štandardu CAdES by mal obahovať len identifikáciu podpisového certifikátu.

Atribút complete-certificate-references

Jedná sa o nepodpisovaný atribút. Atribút obsahuje zoznam štruktúr OtherCertID, ktoré identifikujú certifikát na základe hash hodnoty a prípadne vydavateľa a sériového čísla certifikátu. Atribút umožňuje použiť pre získanie hash hodnoty ľubovolný hashovací algoritmus. Atribút obsahuje referencie na všetky certifikáty z certifikačnej cesty (neobsahuje však referenciu na podpisový certifikát). V podpise môže existovať jediný atribút tohoto typu. Tento atribút sa môže pridať aj do atribútov s časovými razítkami, ktoré definuje štandard CAdES. Referencie budú slúžiť pri získavaní certfikátov potrebných pre overenie časového razítka v prípade potreby v budúcnosti. Rovnako je do časového razítka možné pridať aj atribúty revocation-references, certificate-values a revocation-values.

Atribút complete-revocation-references

Jedná sa o nepodpisovaný atribút a v podpise môže existovať jediný atribút tohoto typu. Atribút obsahuje zoznam štruktúr `CRL_OCSP_Ref`, ktoré identifikujú zoznam revokačných dokumentov pre certifikát. Atribút by mal obsahovať referenciu na revokačné dáta pre podpisový certifikát, nasledovanú revokačnými referenciami pre certifikáty uvedené v atribúte complete-certificate-refs. Poradie revokačných referencií ma byť rovnaké ako poradie referencií na certifikáty. Revokačná referencie identifikuje OCSP odpoveď alebo CRL. V prípade rozdielového CRL je nutné uviesť referenciu na rozdielové CRL a aj na CRL, voči ktorému bolo rozdielové CRL vytvorené. CRL je identifikovaná podľa hashu z DER kódovania celého CRL. Je možné pridať aj identifikáciu CRL podľa vydavateľa, dátumu vydania a poradového čisla CRL, ktorá zjednodušuje dohľadanie CRL. OCSP odpoveď je identifikovaná podľa hashu z DER kódovania OCSP odpovede, prípadne podľa ID odpovede a času vydania.

Atribút certificate-values

Jedná sa o nepodpisovaný atribút a v podpise môže existovať jediný atribút tohoto typu. Atribút by mal obsahovať všetky certifikáty referencované v atribúte complete-certificate-references.

Atribút revocation-values

Jedná sa o nepodpisovaný atribút a v podpise môže existovať jediný atribút tohoto typu. Atribút by mal obsahovať všetky revokačné dokumenty referencované v atribúte complete-revocation-references.

Atribút signature-time-stamp

Jedná sa o nepodpisovaný atribút. Atribút tvorí časové razítko, ktoré má v poli messageImprint v štruktúre TimeStampToken hash hodnotu z reťazca:

- OCTETSTRING z poľa singatureValue v štruktúre SignerInfo

Elektronický podpis môže obsahovať viacero týchto atribútov od rôznych časových autorít.

Atribút CAdES-C-time-stamp

Jedná sa o nepodpisovaný atribút. Atribút dokazuje existenciu validačných referencií v minulosti. Atribút tvorí časové razítko, ktoré ma v poli messageImprint v štruktúre TimeStampToken hash hodnotu z reťazca, ktorý vznikol spojením DER kódovaní (bez kódovania typu a dlžky) týchto objektov:

- OCTETSTRING z poľa singatureValue v štruktúre SignerInfo
- signature-time-stamp
- complete-certificate-references
- complete-revocation-references

Elektronický podpis môže obsahovať viacero týchto atribútov od rôznych časových autorít.

Atribút CAdES-C-time-stamped-certs-crls-references

Jedná sa o nepodpisovaný atribút. Atribút dokazue existenciu validačných referencií v minulosti. Atribút tvorí časové razítko, ktoré ma v poli messageImprint v štruktúre TimeStampToken hash hodnotu z reťazca, ktorý vznikol spojením DER kódovaní (bez kódovania typu a dlžky) týchto objektov:

- complete-certificate-references
- complete-revocation-references

Elektronický podpis môže obsahovať viacero týchto atribútov od rôznych časových autorít.

Atribút archive-time-stamp

Jedná sa o nepodpisovaný atribút. Atribút slúži na predĺženie platnosti elektronické podpisu. Atribút tvorí časové razítko, ktoré ma v poli messageImprint v štruktúre TimeStampToken hash hodnotu z reťazca, ktorý vznikol spojením DER kódovaní týchto objektov:

- encapContentInfo zo štruktúr SingedData
- Certificates a crls položky štruktúry SignedData
- všetky položky štruktúry SignerInfo, včetne podpisovaných a nepodpisovaných atribútov

Elektronický podpis môže obsahovať viacero týchto atribútov.

3.12 Nedostatky štandardu

Aj keď je štandard CAdES pomerne rozsiahly a mnohokrát revidovaný, existujú v ňom nejasné formulácie. Tieto nejasnosti budú pravdepodobne ošetrené v príslušnej politike digitálneho podpisu.

Podpisový certifikát

Aj ked štandard veľmi detailne popisuje umiesnenie certifikátov potrebných pre overenie podpisového certifikátu, menej sa už venuje umiestneniu samotného podpisového certifikátu. Podpisový certifikát je identifikovaný na základe podpisového atribútu Other Signing Certificate resp. ESS Signing Certificate, v atribúte certifikcate-references sa už preto identifácia nenachádza. Atribút certificate-values by mal obsahovat len certifikáte identifikované v certificate-references a preto tam nie je možné pridať podpisový certifikát. Jediné umiestnenie podpisového certifikátu preto zostáva v položke Certificates štruktúry SignedData a takto to definuje aj štandard CMS.

Od podpisu typu CAdES-X Long je nutné pridávať atribút certificate-values a teda zahrnúť do podpisu certifikáty potrebné na overenie podpisového certifikátu. Žiaden z typov podpisov však neprikazuje umiestnenie podpisového certifikátu do poľa Certificates. V praxi tak vytvoríme podpis typu CAdES-Long, ktorý nemusí obsahovať podpisový certifikát.

Tento stav samozrejme nie je žiadúci a vloženie podpisového certfikátu do poľa Certificates by malo byť samozrejmosťou. Knižnica LTES pridáva podpisový certifikát do poľa Certificates už v podpise typu BES. Pri overení tiež predpokladá, že sa v poli Certificates podpisový certifikát nachádza, v opačom pripade skončí overenie chybou.

Zahrnutie podpisovanej správy do podpisu

Štandard CMS umožňuje vytvoriť elektronický podpis, ktorý bude zahrňovať pôvodnú podpísanú správu, ale tiež podpis, ktorý túto správu zarhnutú mať nebude. V prípade. že podpis správu neobsahuje, je nutné správu archivovať na inom mieste.

Pri dlhodobnej archivácií dokumentov nie je praktické archivovať správu a podpis osobitne. Štandard tento problém rieši a prikazuje zahrnúť podpisovanú správu do podpisu už pri definícií podpisu typu BES.

Na inom mieste štandardu CAdES má však pridávanie podpísanej správu skôr odporúčací charakter. Knižnica LTES bude pridávať podpísanú správu do podpisu typu BES.

Viacnásobný výskyt atribútov s časovým razítkom

Štandard pri definícií nepodpisovaných atribútov signature-time-stamp, CAdES-C-time-stamp a CAdES-C-time-stamped-certs-crls-references umožňuje viacnásobné pridanie týchto atribútov do podpisu v prípade viacerých časových autorít.

Situácia, kedy sú potrebné razítka od viacerých časových autorít, môže nastať v prípade, že podpisujúci a overovateľ dôverujú rozdielnym časovým autoritám. Vtedy je potrebné zarhnúť do podpisu razítko od oboch časových autorít. Problém však nastane v prípade pridávania razítka typu CAdES-C-time-stamp. Toto razítko obsahuje hash, pri ktorého výpočte sa používa aj DER kódovanie atribútu signature-time-stamp. Štandard však už nerieši, ako spočítať príslušný hash, ak sa v podpise nachádza viacero atribútov typu signature-time-stamp od rozdielnych časových autorít. Implementácia knižnice LTES tento problém vyriešila zjednodušením. Do podpisu je nie je možné viacnásobné pridanie atribútov s časovými razítkami typu signature-time-stamp, CAdES-C-time-stamp a CAdES-C-time-stamped-certs-crls-references a archívne časové razítka na seba musia naväzovať sériovo. Ak podpisujúci a overovateľ nedôverujú rovnakej časovej autorite, môžu do podpisu vložiť dve štruktúry SignerInfo a tieto podpisy ďalej rozširovať.

Kapitola 4

Analýza implementácie CAdES do OpenSSL

Pred vlastnou impelementáciou štandardu bolo nutné zanalyzovať štandard CAdES s ohľadom na implementáciu do knižnice OpenSSL. V tejto kapitole bude zanalyzovaná samotná knižnica OpenSSL ako aj možnosť rozšírenia knižnice o štadnard CAdES.

Úvod do analýzy implementácie

Implementácia štandardu CAdES si vyžaduje implementovať štandardy, na ktorých je štandard CAdES závislý. Implementácia štandardu CAdES by si preto vyžiadala implementovať tieto funkcionality:

- implementovať kryptografické algoritmy na spočítanie otlačkov dokumentova RSA
- implementovať štrukturý certifikátu, atribútového certifikátu, CRL, OCSP, digitálneho podpisu a časového razítka
- implementovať algoritmy overenie platnosti týchto štruktúr
- implementovať algoritmus vytvorenie digitálneho podpisu
- implementovať nové atribúty, ktoré zavádza štandard CAdES a spôsob ich kódovania do DER, resp. PEM
- implementovať algoritmy vytvorenie a overenie týchto atribútov a tým aj jednotlivých typov podpisu

Implementovať všetky tieto funkcionality je zbytočné. Existujú totiž produkty (napr. GNUtls), ktoré implementujú mechanizmy infraštruktúry PKI. Úlohou tejto práce je implementovať štandard CAdES do knižnice OpenSSL, ktorá prevažnú časť horeuvedených funkcionalít implementuje a je vydávaná ako Open Source.

Analýza open source programu OpenSSL

OpenSSL má za sebou niekoľkoročný vývoj, je bežnou súčasťou stabilných ditribúcií Linux a Unix a pokladá sa za bezpečnú. Implementuje všetky potrebné algoritmy - RSA, SHA, obsahuje modul pre jednoduchú deklarácou ASN1 štruktúr a podporuje štrukrúry X509, CRL a OCSP. OpenSSL v existujúcej podobe umožňuje prácu so štruktúrou elektronického podpisu PKCS#7, poskytuje rozhranie na vytváranie a overovenia podpisu. V súčasnej verzii OpenSSL nepodporuje spracovanie CMS správ. Štruktúra CMS zavádza napr. identifikáciu podpisového certifikátu podľa hash otlačku. Takúto štruktúru však OpenSSL nevie spracovať , OpenSSL teda nedokáže spracovať všetky štruktúry vytvorené podľa štadnardu CMS.

OpenSSL nepodporuje ani prácu s atribútovými certifikátmi - neumožňuje ich vytvárať, overovať a nemá podporu pre štruktúru atribútových certifikátov. Štruktúra CMS však môže obsahovať zoznam atribútových certifikátov, vo verziách 1 alebo 2. OpenSSL atribútové certifikáty nepodporuje, neumožňuje teda ani spracovanie takýchto podpisov. OpenSSL tiež nepodporuje nepriame a rozdielové CRL. Týmito CLR sa preto táto práca ani nezaoberá. Rozhranie funkcií implementácie CAdES by malo byť navrhnuté tak, že prípade budúcej podpory nových typov CRL v OpenSSL, ich bude možné vkladať do podpisu bez nutnosti zmeny rozhrania impelentácie CAdES. OpenSSL nepodporuje ani prácu s časovými razítkami. Časové razítko je CMS správa, ktorá ako podpísanú dáta obsahuje štruktúru TSTInfo. Prácu so štruktúrou TSTInfo OpenSSL neumožňuje.

Exisujú však rozširenia knižnice OpenSSL, ktoré sú distribuované vo forme patch a pridávajú OpenSSL nové funkčnosti. Nakoľko nie sú súčasťou oficiálnej distribúcie, nie sú bežne rozšírené a pre ich použitie treba príslušný patch na zdrojový kód knižnice aplikovať a knižnicu následne skompilovať.

Rozšírenie OpenSSL - OpenTSA

Knižnica OpenSSL v štandardnej distribúcií nemá podporu pre prácu s časovými razítkami. Exituje open source rozšírenie knižnice OpenSSL o podporu časových razítok - OpenTSA. Toto rozširenie je distribuované vo forme patchov. Rozšírnie umožňuje prácu so štruktúrou časového razítka (TSTInfo,Time Stamp Token, Time Stamp Requst a Time Stamp Response), umožňuje vytvorenie a overovanie časového razítka. Rozšírenie OpenTSA do OpenSSL implementuje aj jednoduchú časovú autoritu, ktorá vytvára časové razítka a dokáže ich odoslať cez HTTP protokol. Certifikát časovej autority musí obsahovať v rozšírení certifikátu Rozšírené Použitie Kľúča hodnotu id-smime-aa-signingCertificate, ktorá takýto certifikát umožňuje použiť ako certifikát časovej autority. Rozšírenie OpenTSA zavádza tento typ použitia certifikátu a tiež ho overuje (neumožní použiť ako certifikát ćasovej autority certifikát bez tohoto rozšírenia). Umožňuje tiež v príkazovej riadke OpenSSL vytvárať certifikáty časovej autority obsahujúce príslušné rozšírenie.

Na Internete sú dostupné patche pre rôzne verzie OpenSSL. V čase písania tejto práce bol najnovší patch dostupný pre OpenSSL vo verzií 0.9.8c. Pre ďalšie verzie OpenSSL patche nie sú vydávané, nakoľko sa počíta so zaradenim OpenTSA priamo do oficiálnej distribúcie OpenSSL vo verzii 0.9.9.

Rozšírenie OpenSSL - OpenPMI

Knižnica OpenSSL v štandardnej distribúcií nemá podporu pre prácu s atribútovými certifikátmi. Existuje rozšírenie OpenPMI, ktoré podporu atribútových certifikátov do knižnice pridáva. Toto rozšírenie je podobne ako OpenTSA distribuované vo forme patchov.

Toto rozšírenie implementuje štruktúry atribútových certifikátov vo verzii 2 a dokáže tieto certifikáty aj vytvárať. Neumožňuje však atribútové certifikáty overovať ani voči autorite, ktorá ich vydala, a ani voči certifikátom verejného kľúča, na ktoré sa môžu vzťahovať. Rozširenie tiež nezavádza jednoduché vstupno/výstupné funkcie na prácu s atribútovými certifikátmi, ako je to pre iné štruktúry, ktoré sú súčasťou OpenSSL zvykom.

Z pohľadu tejto práce sa však dá za najväčšie negatívum považovať, že aj napriek tomu, že OpenPMI zavádza do OpenSSL atribútové certifikáty, nezavádza už podporu atribútových certifikátov do iných štruktúr OpenSSL - aj po aplikácií tohoto rozšírenia sa nedajú vkladať atribútové certifikáty do CMS správy. Tento prístup vývojárov OpenPMI je pochopiteľný, nepoznajú PKCS7 modul, ktorý je súčasťou OpenSSL a jeho úprava pre nich nebola zaujímavá.

Pridávanie atribútových certifikátov do štruktúry CMS umožní až nová verzia knižnice OpenSSL 0.9.9. V tejto verzií OpenSSL je už rozšírenie OpenPMI použíteľné a predstavuje zaujímavý spôsob rozšírenia podpory štandardu.

OpenSSL vo verzií 0.9.8c

Vybrať OpenSSL ako základ pre implementáciu štandardu CAdES sa ukázalo ako výhodné. Rozšírenie OpenTSA bude do OpenSSL aplikované a nebude nutné implementovať podporu časových razítok a autority časových razítok. OpenSSL pre implementáciu štandardu CAdES bude vo verzii 0.9.8c, pretože pre túto verziu existuje v súčasnosti najnovší patch OpenTSA. Rozšírenie OpenPMI nebude aplikované, nakoľko neumožňuje pridávanie atribútových certifikátov do CMS správ a teda z pohľadu tejto práce nič pozitívne neprináša. Štandard CAdES však bude impementovaný spôsobom, aby bola prípadná úprava o podporu atribútových certifikátov jednoduchá.

Rozhodnutie nepodporovať atribútové certifikáty znamená, že štandard CAdES, ktorý s nimi počíta, nebude implementovaný úplne. Z praktického hľadiska však absencia atribútových certifikátov nie je kritická. Atribútové certifikáty ako súčasť CMS správ využívajú najmä časové autority, ktoré nimi garantujú maximálnu odchýlku hodín.

Implementácia CAdES nebude podporovať overovanie certifikátov na základe OCSP odpovedí. Implementácia bude používať mechanizmus overovania certifikátov implmementovaný v OpenSSL, ktorý túto funkčnosť neposkytuje.

4.1 Možnosti rozšírenia OpenSSL o štandard CAdES, návrh implementácie

Implementácia štandardu CAdES bude nazvaná LTES (Long TErm Signature). Štandard CAdES je možné implementovať dvoma spôsobmi

1. rozšíriť knižnicu OpenSSL o modul LTES, jednalo by sa teda o rozšírenie knižnice OpenSSL

2. knižnicu LTES implementovať samostatne, knižnica by sa linkovala s OpenSSL

Priama úprava knižnice OpenSSL by umožnila jednoduchšiu distribúciu implementácie LTES. Modul LTES by sa do OpenSSL pridával vo forme patch, podobne ako modul pre časové razítka, neskôr by sa LTES mohlo stať priamo súčasťou oficiálnej distribúcie OpenSSL. Vzhľadom na proces rozširovania distribúcie OpenSSL je však v tejto fáze začlenenie LTES do OpenSSL málo pravdespodobné. LTES teda nebude zatiaľ priamo zahrnutá do knižnice, ale jej prípadné pozdejšie začlenenie by predstavovalo zaujímavý ďalší rozvoj tejto práce.

LTES bude implementovaná ako samostatná knižnica, ktorá sa bude linkovať s knižnicou OpenSSL (s modulom OpenTSA). Knižnica však bude využívať interné mechanizmy knižnica OpenSSL pre prácu s chybovými hláškami a podobne. Tým sa zjednoduší prípadná neskoršia implementácia LTES priamo do knižnice OpenSSL. Nové verzie knižnice OpenSSL obvykle nemenia poskytnuté rozhranie a preto by LTES mala byť kompatibilná aj s novými verziami OpenSSL.

Knižnica LTES bude implementovaná samostatne, bude sa však linkovať s knižnicou OpenSSL. LTES bude zostavená ako zdieľaná knižnica (prípona so) pod opereračným systémom Linux a ako statický linkovaná knižnica (prípona lib) pod operačným systémom Windows. Súčasťou knižnice bude tiež riadková utilita, ktorá bude sprístupňovať základné funkčnosti knižnice z príkazovej riadky.

Všetky typy podpisu (okrem podpisu EPES), ktoré štandard CAdES popisuje, budú knižnicou LTES implementované. Knižnica bude implementovať funkcie na vytvorenie a overenie jednotlivých typov podpisov.

4.2 Licencia OpenSSL

Spôsob nakladania so software určuje licencia, pod ktorou je daný software vydávaný. Licencia, ktorá sa vzťahuje na OpenSSL je daná históriou. Pôvodne vznikol balík kryptografických funkcií SSLeay, ktorý zavádzal vlastnú licenciu. Neskôr bol tento balík zahrnutý do OpenSSL, pre ktorú opäť vznikla nová licencia, na pôvodný balík sa však stále vzťahuje pôvodná licencia SSLeay. Knižnica OpenSSL je takto vydávaná pod duálnou licenciou - aplikujú sa OpenSSL licence a tiež pôvodná SSLeay licence.

Obe licenciu sú tzv. štýlu BSD Open Source. Je jednou z naslobodnejších licencií pre slobodný software, vyžaduje len uvedenie autora a informácií o licencii, spolu s upozornením na zrieknutie sa zodpovednosti za dielo. Licencie OpenSSL a SSLeay naviac zavádzajú povinnosť zmieniť sa o OpenSSL v reklamných materiáloch produkutov založených na OpenSSL a ochranu značky OpenSSL.

Licencia OpenSSL patrí medzi naslobodnejšie a možnému rozširovaniu OpenSSL a následnej distribúcií takéhoto software nestoja v ceste žiadne prekážky.

Kapitola 5

Analýza implementácie CAdES a popis implementácia

Cieľom tejto práce je implementovať všetky typy podpisov. Je preto nutné implementovať podporu všetkých podpisovaných a nepodpisovaných atribútov, ktoré jednotlivé druhy podpisov vyžadujú. Implementované atribúty vymenúva tabuľka 5.1. Ďalšie typy atribútov (napr. Signer Location) nebudú súčasťou imeplementácie štandardu.

Atribut	**Typ podpisu**
Other Signing Certificate	CAdES-BES
ESS Signing Certificate	CAdES-BES
Complete Certificate References	CAdES-C
Complete Revocation References	CAdES-C
Certificate Values	CAdES-X Long
Revocation Values	CAdES-X Long
signature-time-stamp	CAdES-T
CAdES-C-timestamp	CAdES-X Type 1
Certificate-CRL-timestamp	CAdES-X Type 2
archive-time-stamp	CAdES-Archive

Tabuľka 5.1: Tabuľka uvádza atribúty definované štandardom CAdES, ktoré budú implementované

Implementácia atribútov znamenala implementovať všetky typy ASN1 štruktúr, ktoré definuje štandard CAdES. Pre tieto štruktúry bolo potom nutné implementovať funkcie, ktoré zabezpečia kódovania a dekódovanie do kódovania DER. Tiež bolo nutné implementovať funkcie na inicializáciu, uvoľnenie a kopírovanie týchto štruktúr v pamäti.

Súčasťou implementácie bolo vytvorenie funkcií na pridanie nových štruktúr ako podpisovaných a nepodpisovaných atribútov do štruktúry CMS, ako aj definovanie OID identifikátorov objektov, pod ktorými sú atribúty do

CMS pridávané. Štandard CAdES definuje, ako majú byť inicializované jednotlivé štukúry. Pri implementácií boli vytvorené funkcie, ktoré inicializujú štruktúry podľa štandardu a umožnia tiež overenie, či štruktúry zodpovedajú štandardu. Návrh implementácia bol vytvorený s ohľadom na existujúce funkčnosti OpenSSL, pri implementácií sa však aj tak narazilo na množstvo detailov, ktoré si vyžiadali netriviálne riešenia.

V tejto kapitole bude popísaný postup implementácie do zodpovedajúcej hĺbky.

5.1 Osnova implementácie

Pred implementáciou bolo nutné knižnicu OpenSSL zanalyzovať. Analýza sa najprv týkala obecných funkčností ako práca s digitálnym podpisom, overovanie certifikátu a pod. Nasledovala analýza implementácie jednotlivých funkčností a mechanizmov. Po tejto analýze bol zostavený návrh implementácie, ktorý zohľadňoval existujúce funkčnosti knižnice OpenSSL.

OpenSSL implementuje mnoho štandardov, málokedy však implementuje štandard úplne. Knižnica LTES bude využívať existujúcu implementáciu v čo najväčšej miere, aj keď nie je úplná. Alternatívou by totiž bola presná implementácia štandardov knižnicou LTES, čo však nie je predmetom tejto práce. Knižnica OpenSSL sa však stále vyvíja a preto sa dá predpokladať, že v budúcnosti bude implementácia jednotlivých štandardov v OpenSSL dotiahnutá do konca. Nasleduje osnova, ktorá zachytáva postup implementácie knižnice LTES. Jednotlivé body osnovy budú popísané detailnejšie v ďalších kapitolách.

Osnova implementácie CAdES štandardov do knižnice OpenSSL:

1. analýza implementácie OpenSSL (overovanie certifikátu, kódovanie štruktúr, mechanizmu polí a chybových správ)
2. deklarácia ASN1 štruktúr, funkcií pre kódovanie a dekódovanie štruktúr
3. deklarácia NID objektov
4. funkcie na pridanie atribútu do podpisu a funkcie na získanie atribútu z podpisu
5. deklarácia funkcií pre atribút OtherSignHash
6. funkcie na spočítanie hash otlačku pre jednotlivé druhy časových razítok
7. funkcie na vytvorenie žiadostí o časové razítka a pridávanie časových razítok do podpisu
8. funkcie na pridávanie validačných referencií na validačné dáta, spôsob získania validačných dát

9. funkcie na pridanie validačných dokumentov do časového razítka
10. overenie atribútu OtherSignHash a ESSSigningCert
11. overenie časových razítok v podpise
12. implementácia funkcií pre prácu s typom ASN1_GENERALIZEDTIME
13. overenie atribútu `archive-time-stamp`
14. overenie certifikačnej cesty voči validačným referenciám
15. overenie celého podpisu
16. testovanie knižnice LTES
17. riadková utlitia LTES

5.2 Popis mechanizmov OpenSSL

V nasledujúcej kapitole budú popísane mechanizmy, ktoré OpenSSL podporuje. Knižnica LTES tieto mechanizmy používa.

Alokovanie, uvoľnovanie a kopírovanie štruktúr v pamäti

Pre každú štruktúru PKI v OpenSSL je možné pomocou makra definovať sadu funkcií

- new - alokuje štruktúru
- free - uvolní štruktúru z pamäte
- dup - vytvorí kópiu štruktúry a vráti ukazateľ na túto kópiu

Vzhľadom na to, že jazyk C neobsahuje garbage collector sa musí s ukazateľmi pracovať opatrne. Niektoré funkcie OpenSSL typu GET (vrátia položku štruktúry) vrátia iba ukazateľ, niektoré štruktúru skopírujú a vrátia ukazateľ na túto kópiu. Podobne funkcie typu SET niekedy zapíšu priamo ukazateľ, niekedy vytvoria kópiu a zapíšu až ukazateľ na túto kópiu. Nesprávne použitie funkcií OpenSSL môže teda spôsobovať chyby v používaní pamäte.

OpenSSL pri štruktúrach, pri ktorých očakáva časté kopírovanie implementuje tzv. reference-counting. Súčasťou šturktúry je potom čítač, ktorý obsahuje počet referencií na danú štruktúru. Referencie sa rušia volaním príslušnej funkcie free a až keď počet referencií klesne na nulu, je objekt z pamäte uvoľnený. Štruktúre pri každej požidavke na kopírovanie zväčší

```
#define sk_LTES_CRL_VALIDATED_ID_new_null() SKM_sk_new_null(LTES_CRL_VALIDATED_ID)
#define sk_LTES_CRL_VALIDATED_ID_free(st) SKM_sk_free(LTES_CRL_VALIDATED_ID, (st))
#define sk_LTES_CRL_VALIDATED_ID_value(st, i) SKM_sk_value(LTES_CRL_VALIDATED_ID, (st), (i))
#define sk_LTES_CRL_VALIDATED_ID_push(st, val) SKM_sk_push(LTES_CRL_VALIDATED_ID, (st), (val))
#define sk_LTES_CRL_VALIDATED_ID_find(st, val) SKM_sk_find(LTES_CRL_VALIDATED_ID, (st), (val))
#define sk_LTES_CRL_VALIDATED_ID_delete(st, i) SKM_sk_delete(LTES_CRL_VALIDATED_ID, (st), (i))
#define sk_LTES_CRL_VALIDATED_ID_shift(st) SKM_sk_shift(LTES_CRL_VALIDATED_ID, (st))
```

Príklad 5.1: Ukážka makier pre prácu s poľom

počet referencií. Reference couting zabraňuje častému kopírovaniu štruktúr v pamäti a tým šetrí výpočetný výkon počítača. Rerefence-counting je implementovaný u najpoužívanejších štruktúr ako certifikát (X509), CRL (X509_CRL) alebo privátny kľúč (EVP_PKEY).

Z popisu mechanizmu vidno, že riziko zlej práce s pamäťou je veľké. Je preto dôležité poznať zdrojový kód OpenSSL a v závere otestovať knižnicu LTES pomocou nástroja odhaľujúceho memory leaks.

Mechanizmus pre prácu s poľami

Jazyk C nie je objektový jazyk a nie je k nemu dodávaný žiaden framework, ktorý by uľahčoval časté operácie, ako je tomu napríklad u C# a frameworku .NET a pod. Takéto riešenie si preto zvykne každý väčší software implementovať sám. Práca s poľami patrí medzi najčastejšie opakujúce sa práce a OpenSSL implementuje interný mechanizmus pre prácu s nimi. Tento mechanizmus OpenSSL pre prácu s poľami tiež spolupracuje s ASN modulom a umožňuje vytváranie ASN1 štruktúr typu SEQUENCE OF. Mechanizmus je dobre otestovaný, rokmi zdokonaľovaný a jeho použitie prináša mnoho výhod.

OpenSSL deklaruje pole pomocou štruktúry STACK a definuje funkcie ako new, delete, insert, find,sort a pod. Mechanizmus polí pôvodne v OpenSSL neriešil typovú kontrolu ale neskôr bol rozšírený o tzv. modul safestack. Tento modul rieši typovú kontrolu polí pre typy, ktoré sú určené makrom DECLARE_STACK_OF(typ). Keď je toto makro deklarované, vytvorí Perl skript mkstack deklarácie makier na bezpečnú prácu s poľom pre príslušný typ. Práca s mechanizmom polí v OpenSSL sa tak zjednoduší a nie je zdrojom potencionálnych chýb.

Modul safestack bude použitý aj pri implementácií knižnice LTES. Upravený skript mkstack vytvorí makrá pre prácu s poľom. Využite tohto mechanizmu tiež zjednoduší prípadnú pozdejšiu integráciu modulu LTES do OpenSSL. Príklad 5.1 uvádza vytvorené makrá na prácu s poľom štruktúry OTHER_SIGN_CERT_ID.

Kódovanie a dekódovanie objektov do formátu DER

Jednou z najdôležitejších činností pri spracovaní PKI štruktúr je kódovanie a dekódovanie štruktúry do formátu DER. Túto činnosť OpenSSL zautoma-

```
unsigned char *out = malloc( size );
i2d\_PKCS7(p7,out);
```

Príklad 5.2: Implementácia dekódovania ASN1 štruktúry)

tizovalo a definuje pre každú ASN1 štruktúru, ktorý je v OpenSSL implementovaný, dvojicu funkcií d2i a i2d so suffixom označujúcim meno štuktúry (napr. d2i_PKC7 a i2d_PKCS7).

Fukcie majú podobnú syntax, líšia sa len v štruktúre, ktorú kódujú. V nasledujúcom príklade budú popísané funkcie kódujúce štrktúru PKCS7.

Funkcia pre kódovanie objektu PKCS7 má nesledovnú syntax:

```
int i2d_PKCS7( PKC7* pkcs7,unsigned char *out)
```

Funkcia príjma ako parameter štruktúra a pole znakov, do ktorého zapíše kódovaný reťazec. Funkcia vracia počet bytov. Ak sa ako ukazateľ na pole znakov zadá NULL, funkcia vráti dĺžku kódovaného reťazca. Typicky sa teda najskôr zavolá funkcia pre zistenie dĺžky reťazca, v ďalšom kroku sa pole potrebnej dĺžky alokuje a druhým volaním funkcie sa do poľa zapíše kódovnaý reťazec. Tento postup ukazuje príklad 5.2.

Funkcia na dekódovanie ma nasledovnú syntax:

```
PKCS7 *d2i_PKCS7(PKC7** pkcs7,unsigned char** in,long len)
```

Funkcia príjma ako parameter referenciu na ukazateľ na existujúcu štruktúru, referenciu na ukazatel pola znakov a dĺžku tohoto poľa. Funkcia dekóduje príslušné pole dát a vráti ukazateľ na štruktúru. V prípade, že je prvý parameter nie je NULL, použije sa táto štruktúra na inicializáciu návratovej štruktúrz, v opačnom prípade sa alokuje nová štruktúra.

Vyššie uvedené funkcie kódujú jednoduché štruktúry. V prípade, že je potrebné kódovať pole (sekvenciu) štruktúr, je potrebné použiť inú dvojicu funkcií. Pre kódovanie sekvencií existujú v OpenSSL nasledujúce funkcie:

```
STACK *ASN1_seq_unpack(unsigned char *buf, int len,
d2i_of_void *d2i, void (*free_func)(void *))
```

Funkcia príjma ako parametre ukazatel na pole dát (DER kód sekvencie), dĺžku tohoto poľa a ukazatele na dve funkcie. Prvá funkcia dekóduje z DER jednotlivé štruktúry a druhá funkcia uvoľnuje štruktúry z pamäte. Funkcia vráti pole štruktúr, ktoré je následne nutné vhodne pretypovať.

```
unsigned char *ASN1_seq_pack(STACK *safes, i2d_of_void *i2d,
unsigned char **buf, int *len)
```

Funkcia príjma ako parametre pole štruktúr, funkciu pre kódovanie jed-

notlivých štruktúr do DER, referenciu na ukazateľ na poľa znakov a referenciu na typ int. Funkcia vráti ukazateľ na pole znakov, ktoré tvorí DER kódovanie sekvencie. Do posledného parametra zapíše dĺžku tohto poľa. Ak je zadaná referencia na pole dát, sekvencia za zákoduje do tohoto poľa, inak sa alokuje nové pole.

Mechanizmus chybových správ

OpenSSL implementuje mechanizmus spracovania chybových správ. Zavádza dátovú štruktúru (cyklické pole) a funkcie na prácu s týmto poľom. OpenSSL definuje funkcie na pridávania chybových správ do štruktúry, ich mazanie a výpis tejto štruktúry. Pre každý modul, ktorý je súčasťou knižnice OpenSSL (X509, PKCS7, ASN1) je definovaná funkcia, ktorá pridáva vzniknuté chyby do internej dátovej štruktúry. Vhodným spracovaním chybových stavov je možné poskytnúť užívateľovi zoznam funkcií a chýb, ako chyba postupne eskalovala až do užívateľom volanej funkcie.

Záznam o chybe sa skladá z informácie o názov modulu (X509, PKCS#7 atď.), názve funkcie, názve súboru a číslo riadku, kde chyba vznikla. Položka tiež obsahuje dôvod vzniku chyby. Modul, funkcia a dôvod chyby sú textové reťazce, v štruktúre chyby sú však identifikované kódmi priradenými príslušnému modulu, funkcii a dôvodu. Kódy sú priradzované manuálne v hlavičkovom súbore príslušného modulu. Pri výpise chyby sa ku príslušným kódom priradia textové reťazce, ktoré sú potom zobrazené na výstupe.

Modul implementujúci dlhodobý podpis bude využívať rovnaký mechanizmu ako OpenSLL a bude chyby vkladať do rovnakej štruktúry. Tým naviaže na existujúce riešenie a nebude nutné implementovať nové riešenie spracovnania chýb. Pre implementáciu mechanizmu chýb podľa OpenSSL je nutné definovať príslušné kódy funkcií a dôvodov chýb a následne k nim priradiť textové správy. Toto priradenia bude definované v poli, pomocou ktorého budú chybové hlášky modulu LTES prekladané.

Definovať takéto pole ručne by bolo veľmi pracné a bol by to potencionálny zdroj chýb. Pri definovaní týchto polí preto bude použitý upravený Perl skript mkerr.pl, ktorý je súčasťou OpenSSL a zo zdrojových súborov modulu vytvorí príslušné polia chýb.

Použitie tohoto skriptu definuje polia s chybovými stavmi podľa zvyklostí OpenSSL a pozdejšie prípadné integrovanie modulu do OpenSSL nebude musieť riešiť mechanizmus chýb. Pri požiadavke na integráciu modulu do OpenSSL by zostávalo len zahrnúť modul LTES do internej kolekcie modulov OpenSSL.

Implementovanie mechanizmu chýb podľa vzoru OpenSSL umožnilo jednoduché spracovanie chýb. Každé volanie funkcie preto zachytáva chybové stavy, na základe ktorých generuje prehľadné chybové hlásenie. Pri chybe v knižnici LTES je funkcia ukončená chybovým návratovým kódom, aby mohla volajúca funkcia na túto chybu reagovať. Takýmto postupom je možné chybu propagovať až do užívateľom volanej funkcie. Zásobník s chybovými správ-

```
2292:error:8007500B:lib(128)
:LTES_DATAVERIFY:X509 lib:.\ltes_verify.c:1187:Verify error:CRL has expired
```

Príklad 5.3: Ilustrácia chybových správ v LTES)

vami je potom možné vypísať volaním funkcie `ERR_print_errors`. Príklad 5.3 ukazuje príklad takéhoto chybového výstupu knižnice LTES.

Práca s podpisovanými a nepodpisovanými atribútmi CMS

Modul PKCS7, ktorý je súčasťou OpenSSL umožňuje pracovať so štruktúrou podpisu CMS SignerInfo. Táto štruktúra mimo iné obsahuje aj zoznam podpisovaných a nepodpisovaných atribútov, ktoré sú súčasťou podpisu. OpenSSL definuje funkcie na prácu so zoznamom podpisovaných a nepodpisovaných atribútov, pridávanie a mazanie atribútov zo zoznamu.

Implementácia funkcií na pridávanie podpisovaných a nepodpisovaných atribútov modulu PKCS7 však obsahuje kontrolu na duplicitu typu atribútov s rovnakým identifikátorom. Pomocou funkcí modulu PKCS7 preto nie je možné pridávať viac atribútov pod tym istým OID.[1]

Nepodpisovaný atribút archive-timestamp je všal podľa štandardu CAdES možné pridať do zoznamu atribútov opakovane (a tým predlžovať platnosť elektronického podpisu). Viacnásobné pridanie tohto atribútu nie je možné realizovať existujúcim rozhraním OpenSSL a preto je súčasťou modulu LTES aj funkcie na viacnásobné pridanie tohoto atribútu do zoznamu nepodpisovaných atribútov.

Statické premenné OpenSSL

OpenSSL zavádza mnoho statických premenných pre prácu s hash algoritmami, chybovými správami a podobne. Tieto premenné sú inicializované pred ich prvým použitím. Na záver práce je dobré zas tieto premenné uvoľniť, aby zbytočne nezaberali pamäť. Knižnica LTES inicializuje všetky potrebé statické premenné vo funkcií `LTES_Init`, ktorá by mala byť zavolaná pred volaním funkcií knižnice LTES. Po skončení práce s knižnicou LTES by mala byť zavolaná funkcia `LTES_Exit`, ktorá uvoľní statické premenné z pamäti.

Overenie certifikátu v OpenSSL

Algoritmus overenie certifikátu spĺňa požiadavky štandardu RFC 5280. Algoritmus konštruuje certifikačnú cestu, ktorá musí byť zakončená dôveryhodnou kotvou. Následne sú overené všetky certifikáty certifikačnej cesty voči CRL a na záver prebehne kontrola spôsobu použita certifikátov.

[1] Pomocou funkcií, ktoré sú súčasťou OpenSSL tak nie je možné pridať viac krát napríklad atribút typu archive-time-stamp.

Vstupom algoritmu je overovaný certifikát a tzv. verifikačný kontext. Tent kontext obsahuje

- zoznam dôveryhodných a nedôveryhodných certifikátov
- zoznam umiestnení adresárov s certifikátmi (adresáre v súborovom systéme)
- zoznam CRL
- čas, ku ktorému sa má certifikát overovať
- rôzne príznaky, ktoré špecifikujú postup overovania

Algoritmus najprv zostaví certifikačnú cestu. [2] Certifikáty sú pri konštrukcií cesty vyhľadávané na základe vydavateľa, pri konštrukcii certifikačnej cesty sa totiž začína overovaným certifikátom a hľadá sa vždy certifikát vydavateľa. Certifikát sa hľadá v zozname certifikátov, prehľadávajú sa však aj adresáre na disku.[3] Pri hľadaní certifikátu sa zohľadňuje aj zhoda polí Identifikácia kľúča predmetu a Identifikácia kľúča úradu.

V ďalšom kroku overí algoritmus platnosť jednotlivých certifikátov na základe CRL a doby platnosti samotného certifikátu. Pri overovaní certifikátu voči CRL sa kontroluje aj platnosť CRL, ktorá je dáná hodnotou thisUpdate, nextUpdate v štruktúre CRL. Nakoniec sa overí spôsob použita certifikátu - napríklad či sa jedná o certifikát časovej autority a podobne.

Algoritmus overovania certifikátov má však aj niekoľko nedostatkov. Medzi najväčšie nedostatky mechanizmu overovania certifikátu určite patrí, že pri overovaní nepodporuje OCSP protokol. OpenSSL tento protokol implementuje, pri overovaní certifikátu však nie je zohľadnený.

Ďalším nedostatkom mechanizmu je, že nedokáže do zoznamu CRL načítať dve CRL od toho istého vydavateľa. Na overovateľovi tak zostáva vyhľadanie správneho CRL. Algoritmus tiež nepodporuje vyhľadanie CRL podľa Identifikácie kľúča úradu. Ďalším nedostatkom je hľadanie certifikátov v súborovom systéme. Adresáre s certifikátmi sú prehľadávané až pri požiadavke na konkrétny certifikát, nie je preto možné výber certifikátu ovplyvniť.

OpenSSL však poskytuje možnosť implementovať vlastné funkcie pre vyhľadávanie certifikátu a nastaviť verifikačný kontext tak, aby volal uživateľom definováne funkcie a nie predvolené funkcie. Tento mechanizmus používa aj knižnica LTES, keď nájdené certifikáty naviac kontroluje voči validačným referenciám uloženým v podpise.

Pri implementácií knižnice LTES bol použitý mechanizmus overovania certifikátu z OpenSSL aj napriek týmto nedostatkom. Alternatívou totiž bolo implementovanie vlastného mechanizmu overovanie certifikátu, čo však bolo nad rámec stanovených cieľov. Existuje tiež predpoklad, že v ďalších verziách

[2] Certifikačná cesta môže obsahovať dôvyhodné aj nedôveryhodné certifikáty, musí však byť zakončená dôveryhodným samopodpísaným certifikátom.

[3] Adresáre s certifikátmi sú určené vo verifikačnom kontexte.

OpenSSL bude mechanizmus overovania certifikátu dokonalejší. Vlastná implementácia overovania certifikátu je preto z dlhodobého pohľadu zbytočná. Naviac problémy mechanizmu overenia nie sú natoľko závažné, aby sa nedal použíť už v existujúcej forme.

5.3 Identifikátory OID a deklarácia NID objektov

Štadardy PKI používajú na identifikáciu objektov tzv. OID identifikátory. Knižnica OpenSSL vytovorila podporu pre OID identifikátory a definuje dátový typ ASN1_OBJECT. Definuje tiež funkcie pre kódovanie do štruktúry DER formátu.

OpenSSL definuje tabuľku objektov. Príslušný objekt je v nej identifikovaný celočíselným identifikátorom určujúcim index objektu v tabuľke - tzv NID číslom. Pod týmto NID číslom sa v tabuľke nachádza samotný objekt, ktorý pozostáva z OID identikátora a názvu objketu. OpenSSL definuje funkcie, ktoré umožňujú pridávať do tejto tabuľky ďalšie objekty.

V OpenSSL je možné aj nové OID, ktoré nie sú definované v OpenSSL, identifikovať pomocou NID čisel, vyhľadávať v zoznamoch atribútov na základe NID čísel a podobne. Práca s NID číslami a celočíselnými identifikátormi je z pohľadu jazyka C jednoduchšia, ako práca so samotnými ASN1 objektami.

LTES zavádza 11 nových objektov, pre ktoré definuje NID čisla. Do tabuľky objektov pridáva nové objekty volaním funkcie `OBJ_create`, ktorá príjma OID nového objektu a dva názvy pre jednoduchšiu identifikáciu objektu. Nové NID čisla sú uložené v statických premenných knižnice LTES a sú definované vo funkcii `LTES_Init`, ktorá inicializuje statické premenné knižnice. Na konci práci s knžnicou LTES je možné pridané objekty uvoľniť volaním funkcie `OBJ_cleanup`, ktorú knižnica LTES volá vo funkcií `LTES_Exit`.

Úplný výpis objektov je v nasledujúcej tabuľke 5.2.

5.4 Deklarácia typov pre ASN1 štruktúry

OpenSSL obsahuje podporu pre jednoduche vytváranie nových ASN1 štruktúr typu CHOICE a SEQUENCE. LTES vyžadovalo implementovať mnoho nových štruktúr a knižnica OpenSSL tento proces uľahčila.

Prvým krokom pri definovaní nových štruktúr je definovanie štruktúr v jazyku C, na ktoré sa budú nové ASN1 štruktúry neskôr mapovať. Nová štruktúra v jazyku C, ktorá zodpovedá ASN1 štruktúre, musí obsahovať ukazatele na všetky zložky štruktúry v jazyku ASN1. Deklarácia bude podrobnejšie popísaná u zložených typov štruktúr (SEQUENCE, SEQUENCE OF a CHOICE), ktoré boli použité v knižnici LTES.

NID identifikátor	OID objektu	Atribút určený objektom
NID_ets_otherSigCert	1.2.840.113549.1.9.16.2.19	OtherSignCert
NID_signatureTST	1.2.840.113549.1.9.16.2.14	signatureTST
NID_certificateRefs	1.2.840.113549.1.9.16.2.21	certificate References
NID_revocationRefs	1.2.840.113549.1.9.16.2.22	revocation References
NID_certValues	1.2.840.113549.1.9.16.2.23	atribút certificate Values
NID_crlValues	1.2.840.113549.1.9.16.2.24	revocation Values
NID_escTimeStamp	1.2.840.113549.1.9.16.2.25	escTimeStamp v podpise Type1
NID_certCRLTimestamp	1.2.840.113549.1.9.16.2.26	certCRLTimeStamp v pod. Type2
NID_archiveTimeStamp	1.2.840.113549.1.9.16.2.27	archive Timestamp
NID_attrCertificateRefs	1.2.840.113549.1.9.16.2.44	certificate References
NID_attrRevocationRefs	1.2.840.113549.1.9.16.2.45	revocation References

Tabuľka 5.2: Tabuľka vymenúva pridané identifikátory objektov tak, ako ich definuje štadard CAdES

```
CrlIdentifier ::= SEQUENCE {
clrissuer Name,
crlIssuedTime UTCTime,
crlNumber INTEGER OPTIONAL
}

typedef struct LTES_CRL_IDENTIFIER_st
{
X509_NAME *name;
ASN1_UTCTIME *UTCTime;
ASN1_INTEGER *crlNumber;
} LTES_CRL_IDENTIFIER;
```

Príklad 5.4: Deklarovanie ASN1 typu sekvencia)

Jednoduché je deklarovať štruktúru typu SEQUCNCE OF -type-. Takáto štruktúra obsahuje ľubovolný počet položiek určeného typu. Typ SEQUENCE OF je v OpenSSL deklarovaný ako pole (STACK) položiek. Je však nutné deklarovať typ vnútornej položky poľa. Na kódovanie a dekódovanie polí slúžia univerzálne funkcie `ASN1_seq_pack` a `ASN1_seq_unpack`.

Deklarácia sekvencií

V prípade ASN1 sekvencií je C štruktúra vytvorená prepisom jednotlivých zložiek. Typy, z ktorých sekvencia pozostáva, však už musia byť definované ako C štruktúry.

Príklad 5.4 popisuje prepis ASN1 štruktúry CrlIdentifier do štruktúry jazyka C.

```
OtherHash ::= CHOICE {
sha1Hash OtherHashValue, -- this contains a SHA-1 hash
otherHash OherHashAlgAndValue
}

typedef struct LTES_other_hash_st {
int type;
union {
LTES_OTHER_HASH_VALUE *sha1_hash;
LTES_OTHER_HASH_ALG_AND_VALUE *other_hash;
} value;
} LTES_OTHER_HASH;
```

Príklad 5.5: Deklarovanie ASN1 typu výber)

```
typedef struct LTES_CRL_OCSP_REF_st
{
STACK_OF(LTES_CRL_VALIDATED_ID) *clrs; /* [0] */
STACK_OF(LTES_OCSP_RESPONSES_ID) *ocspids; /* [1] */
LTES_OTHER_REV_REFS *otherRev; /* [2] */
} LTES_CRL_OCSP_REF;
```

Príklad 5.6: Deklarovanie ASN1 štruktúry, ktorá obsahuje položku typu SEQUENCE OF)

Deklarácia výberu

ASN1 zložený typ CHOICE opäť vyžaduje, aby zodpovedajúca C štruktúra obsahovala všetky položky. Keďže je však nakoniec vyplnená iba jedna položka, je možné ušetriť pamäťové miesto štruktúry použitím C konštrukcie union. C štruktúra, ktorá zodpovedá typu CHOICE, však musí obsahovať aj celočíselný identifikátor type. Tento identifikátor určuje, ktorá zo zložiek typu CHOICE je platná.

Príklad 5.5 popisuje prepis ASN1 štruktúry OtherHash do štruktúry jazyka C.

Typ SEQUENCE_OF

Ak je položka štruktúry definovaná ako SEQUENCE_OF, táto položka sa deklaruje ako pole objektov a teda typ STACK.

Príklad 5.6 popisuje prepis ASN1 štruktúry crlOcspRef do štruktúry jazyka C.

Voliteľné položky (OPTIONAL) a implicitné položky žiadne špeciálne postupy pri definovaní C štruktúr nevyžadujú, ich význam sa prejaví sa až pri vytváraní mapovania ASN1 štruktúr na C štruktúry.

5.5 Mapovanie ASN1 štruktúr

Druhým krokom pri tvorení nových ASN1 štruktúr je mapovanie existujúcich C štruktúr na ASN1 štruktúry. Jednotlivé zložky ASN1 štruktúr sú

```
ASN1_SEQUENCE(LTES_CRL_IDENTIFIER) = {
ASN1_SIMPLE(LTES_CRL_IDENTIFIER,name,X509_NAME),
ASN1_SIMPLE(LTES_CRL_IDENTIFIER,UTCTime,ASN1_UTCTIME),
ASN1_OPT(LTES_CRL_IDENTIFIER,crlNumber,ASN1_INTEGER)
} ASN1_SEQUENCE_END(LTES_CRL_IDENTIFIER);
```

Príklad 5.7: Mapovanie ASN1 typu SEQUENCE)

```
ASN1_CHOICE(LTES_OTHER_HASH) = {
ASN1_SIMPLE(LTES_OTHER_HASH,value.sha1_hash,LTES_OTHER_HASH_VALUE),
ASN1_SIMPLE(LTES_OTHER_HASH,value.other_hash, LTES_OTHER_HASH_ALG_AND_VALUE)
}
ASN1_CHOICE_END(LTES_OTHER_HASH);
```

Príklad 5.8: Mapovanie ASN1 typu CHOICE)

určené poradím v sekvencií, ale aj niektorým z príznakov EXIPLICIT, IMPLICIT alebo OPTIONAL. OpenSSL definuje súbor makier pre definovanie mapovania, umožňuje mapovať C štruktúry na typ SEQUENCE a CHOICE, podporuje tiež príznaky IMPLICIT a OPTIONAL.

Mapovanie sekvencií

Pri mapovaní sekvencie je dôležite zachovať poradie jednotlivých zložiek. Príklad 5.7 ukazuje mapovanie štruktúry crlIdentifier (v LTES nazvaná LTES_CRL_IDENTIFIER), ktorej deklarácia bola úkazaná v priklade 5.4. Bežné položky su mapované pomocou makra ASN1_SIMPLE, položka z príznakom OPTIONAL je mapovaná pomocou makra ASN1_OPT.

Mapovanie výberu (CHOICE)

Mapovanie typu CHOICE je podobné ako typy SEQUENCE, na poradí položiek však nezáleží. Pri mapovaní CHOICE štruktúr však existuje obmedzenie, ktoré vyplýva už zo samotného štandardu ASN1 a kódovanie DER. Do štruktúry typu CHOICE nie je možné vložiť položky, ktoré sú rovnakého typu. Ich DER kódovanie by totiž začínalo rovnako, a pri dekódovaní by sa nedalo rozhodnúť, ktorá položka je kódovaná. Na odlíšenie položiek, ktorých začiatok kódovanie do DER je rovnaký, sa používa príznak IMPLICIT a takto sú navrhnuté aj ASN1 štruktúry štandardu CAdES.

Príklad 5.8 ukazuje mapovanie štruktúry otherHash.

Mapovanie implicitných zložiek

Pri mapovaní implicitných zložiek sa používa makro ASN1_IMP, pričom je nutné zadať aj číslo tagu, pod ktorým sa bude zložka nachádzať. Z príkladu je vidno, že je možné kombinovať príznak implicitný a voliteľný použitím

```
ASN1_SEQUENCE(LTES_CRL_OCSP_REF) = {
ASN1_IMP_SEQUENCE_OF_OPT(LTES_CRL_OCSP_REF,clrs,LTES_CRL_VALIDATED_ID,0),
ASN1_IMP_SEQUENCE_OF_OPT(LTES_CRL_OCSP_REF,ocspids,LTES_OCSP_RESPONSES_ID,1),
ASN1_IMP_OPT(LTES_CRL_OCSP_REF,otherRev,LTES_OTHER_REV_REFS,2)
}
ASN1_SEQUENCE_END(LTES_CRL_OCSP_REF);
```

Príklad 5.9: Mapovanie ASN1 typu IMPLICIT)

makra ASN1_IMP_OPT. V prípade 5.9 tiež vidno mapovanie zloženého typu SEQUENCE_OF a použitie príslušného makra.

Implementácia podporných metód pre ASN1 štruktúry

Po tom, ako je C štruktúra správne nadeklarovaná a namapovaná na ASN1 štruktúru, je možné vytvoriť funkcie pre kódovanie a dekódovanie štruktúry do formátu DER. Funkcie sa dekl. makrom `DECLARE_ASN1_FUNCTIONS(st)` a definujú makrom `IMPLEMENT_ASN1_FUNCTIONS(st)`, kde sa ako parameter zadá meno štruktúry.

Mechanizmus OpenSSL, ktorým sa vytvárajú príslušné funkcie pre kódovanie a dekódovanie je zložitý. Pri mapovaní štruktúry sa vytvára pole, ktorého jednotlivé zložky popisujú napríklad spôsob kódovnia alebo typ príslušnej zložky štruktúry. Tieto informácie sú potom použité pri požiadavke na kódovanie alebo dekódovanie C štruktúry do formátu DER. Vyššie uvedené makrá tiež vytvoria metódy new, free a dup pre príslušnú C štruktúru.

5.6 ASN1 štrukúry knižnice LTES

Štandard CAdES definuje ASN1 štruktúry, ktoré bolo nutné pri implementácií vytvoriť a namapovať. Boli deklarované všetky štuktúry, ktoré štandard CAdES definuje. Tabuľka 5.3 uvádza zoznam týchto štruktúr.

Štruktúra v C	**ASN1 typ**	**Typ štruktúry**
LTES_OTHER_HASH_ALG_AND_VALUE	OtherHashAlgAndValue	SEQUENCE
LTES_OTHER_HASH	OtherHash	CHOICE
LTES_OTHER_CERT_ID	OtherCertID	SEQUENCE
LTES_OTHER_SIGNING_CERTIFICATE	OtherSigningCertificate	SEQUENCE
LTES_OTHER_REV_REFS	OtherRevRefs	SEQUENCE
LTES_CRL_IDENTIFIER	CrlIdentifier	SEQUENCE
LTES_CRL_VALIDATED_ID	CrlValidatedID	SEQUENCE
LTES_OCSP_IDENTIFIER	OcspIdentifier	SEQUENCE
LTES_OCSP_RESPONSES_ID	OcspResponesID	SEQUENCE
LTES_COMPLETE_REVOCATION_REFS	SEQ OF OtherCertID	SEQUENCE OF
LTES_COMPLETE_CERTIFICATE_REFS	SEQ OF CrlOcspRef	SEQUENCE OF
LTES_ATTRIBUTE_CERTIFICATE_REFS	SEQ OF OtherCertID	SEQUENCE OF
LTES_CRL_OCSP_REF	CrlOcspRef	SEQUENCE
LTES_OTHER_REV_VALS	OtherRevVals	SEQUENCE
LTES_REVOCATION_VALUES	RevocationValues	SEQUENCE

Tabuľka 5.3: Tabuľka uvádza zoznam implementovaných ASN1 štruktúr

5.7 Práca s atribútmi

Štandard CAdES rozširuje štandard elektronický podpis o podpisované a nepodpisované atribúty. Pri implementácií štandardu CAdES bolo nutné zmapovať možnosti OpenSSL pri práci s atribútmi PKCS7.

Z pohľadu knižnice LTES (ale aj OpenSSL) je nutné rozdeliť atribúty do dvoch skupín. Jednoduche atribúty budú také, ktoré obsahujú jednoduchú štruktúru. Zložené atribúty budú také, ktoré obsahujú pole jednoduchých štruktúr. Pre názornosť sú uvedené príklady jednoduchého a zloženého atribútu

- LTES_OTHER_SIGNING_CERT je jednoduchý atribút, tvorí ho jednoduchá štruktúra s dvoma zložkami.
- LTES_CERTIFICATE_VALUES je zložený atribút, je definovaný ako STACK_OF(X509) a tvorí ho pole certifikátov

Pre užívateľa knižnice by malo byť pridávanie a získanie atribútov z podpisu čo najjednoduchšie. LTES preto síce poskytuje užívateľovi funkcie na pridávanie a získavanie atribútov priamo z podpisu, neodporúča sa ich však používať. V ďalšej časti tejto práce totiž budú popísané funkcie, ktoré sa postarajú o správnu inicializáciu atribútov a ich následné pridanie do podpisu, alebo získanie atribútov z podpisu a ich verifikáciu.

Pridávanie atribútov PKCS7

Na pridávania atribútov do štruktúry SignerInfo slúži dvojica funkcií:

- PKCS7_add_signed_attribute(si,nid,type,seq)
- PKCS7_add_attribute(si,nid,type,seq)

Prvá z funkcií pridáva podpisové atribúty a je nutné ju zavolať pre spočítaním podpisu. Druhá funkcia pridáva nepodpisované atribúty.

Parametre týchto funkcií sú:

1. štruktúra SignerInfo, do ktorej má byť atribút pridaný
2. NID identifikátor objektu, pod ktorým má byť atribút pridaný
3. typ atribútu, v LTES boli pridávané iba zložené typy a preto bola použitá hodnota V_ASN1_SEQUENCE
4. atribút v DER kódovaní, v prípade typu atribútu SEQUENCE by mal byť atribút reprezentovaný ako ASN1_STRING.

Postup pri pridávaní atribútu je zavislý od toho, či sa jedná o jednoduchý alebo zložený atribút.

1. Pri pridávaní jednoduchého atribútu sa najskôr atribút zakóduje do DER formátu a DER formát sa prevedie na ASN1_STRING pomocou funkcie ASN1_pack_string. Následne sa zavolá jedna z funkcií na pridanie atribútu do podpisu.

```
revVal = LTES_create_revocation_values(ctx);
seq = ASN1_pack_string(revVal,i2d_LTES_REVOCATION_VALUES,NULL);
PKCS7_add_attribute(si,NID_revocationValues,V_ASN1_SEQUENCE,sq);
```

2. Na pridávanie zložených atribútov bola do knižnice LTES implementovaná funkcia LTES_add_attribute_seq. Tá volá funkciu ASN1_seq_pack, následne získaný DER kód zabalí do typu ASN1_STRING a zavolá funkciu na pridanie atribútu do podpisu.

```
revRefs = LTES_create_revocation_refs(ctx,EVP_sha1());
LTES_add_attribute_seq(si,NID_revocationRefs,revRefs,i2d_OCSP_REF)
```

Špeciálny postup vyžadovalo pridávanie atribútu archívneho časového razítka. Funkcie poskytované knižnicou OpenSSL neumožňujú viacnásobné pridávanie atribútu s rovnakým NID. LTES preto implementuje funkcia

```
int LTES_add_attribute(STACK_OF(X509_ATTRIBUTE) **sk, int nid,
int atrtype, void *value)
```

Rozhranie funkcie je podobné ako PKCS7_add_signed_attribute, avšak namiesto štruktúry SignerInfo príjma ukazateľ na pole atribútov. Tento parameter robí funkciu univerzálnejšou a umožňuje viacnásobné pridávanie podpisovaných aj nepodpisovaných atribútov.

Získavanie atribútov zo štruktúry PKCS7 Získavanie atribútov sa z pohľadu knižice LTES rozdeľuje na dve časti:

1. získanie atribútu z podpisu
2. dekódovanie atribútu

Pre získanie atribútu z podpisu sa použije func. X509at_get_attr_by_NID, ktorá vráti atribút podľa objektu (reprezentovaného NID číslom), pod ktorým je atribút uložený. Funkcia prechádza pole atribútov a vracia index hľadaného atribútu. Funkcia ma nasledovné rozhranie:

```
X509at_get_attr_by_NID(sk,nid,idx)
```

Funkcia ako parameter príjma pole atribútov, identifikátor objektu NID a začiatočný index, od ktorého bude v pole prehľadávať. Funkcia vráti index požadovaného atribútu. V prípade, že sa atribút nepodarí nájsť, funkcia vráti -1. Ako parameter funkcie sa vkladá aj index, od ktorého ma pole nepodpisovaných atribútov prehľadávať. Opakovaným volaním funkcie sa dajú získať

```
attr_pos1 = X509at_get_attr_by_NID(sk,NID_archiveToken,-1); \\
attr_pos2 = X509at_get_attr_by_NID(sk, NID_archiveToken,attr_pos1);\\
```

Príklad 5.10: Spôsob získania viacnásobného atribútu)

```
seq = LTES_get_attribute(sk,nid);
p = seq->data;
return d2i_PKCS7(NULL,&p,seq->length);
```

Príklad 5.11: Príklad ukazuje získanie atribútu)

aj atribúty s rovnakým typom, ako ukazuje príklad 5.10 (archívne časové razítka majú pozície attr_pos1 a attr_pos2). Po získaní indexu požadovaného atribútu sa atribút získa z poľa atribútov volaním funkcie `X509at_get_attr`.

Po získaní atribútu je tento atribút nutné dekódovať z formátu DER. Musí sa rozlišovať, či sa jedná o jednoduchý alebo zložený atribút. LTES vytvorilo pre dekódovanie jednoduchého atribútu funkciu `LTES_get_attribute`, ktorá vráti DER kódovanie atribútu vo forme ASN1_STRING, následne je reťazec dekódovaný d2i funkciou pre príslušný typ. Proces získavania jednoduchého atribútu ukazuje príklad 5.11.

Pre zložený atribút bola implementovaná func. `LTES_get_attr_sequence`. Funkcia po získaní premennej typu ASN1_STRING z atributu (ako v prípade vyššie), zavolá funkciu na dekódovanie sekvencie `ASN1_seq_unpack`.

Funkcie pre získavanie atribútov z podpisu pracujú na nízkej úrovni. Pracujú s DER formátom dát, množstvom príznakov a ukazateľov. Použitie takýchto funkcií môže byť pre bežného užívateľa mätúce. Vo verejnom rozhraní preto LTES definuje funkciu typu GET pre každý z atribútov, ktoré definuje štandard CAdES.

5.8 Vytváranie atribútu Other Signing Certificate

Podpis typu BES vyžaduje zahrnutie jedného z atribútov Other Signing Certificate (otherSignCert) alebo ESS Signing Certificate (ESSSignCert) medzi podpisované atribúty. Knižnice LTES nepodporuje pridávania atribútu ESSSignCert do podpisu, štandard to totiž nevyžaduje (LTES však overuje aj ESSSignCert). Atribút otherSignCert obsahuje hash hodnotu podpisového certifikátu, pričom môže obsahovať aj identifikačné údaje certifikátu (IssuerAndSerial). Tento atribút podporuje rôzne typu hashovacích algoritmov pre výpočet otlačku z certifikátu.

Funkcia LTES_set_other_hash inicializuje štrukt. LTES_OTHER_HASH. Táto štuktúra je použitá aj pri atribúte LTES_CERTIFICATE_REFS, kde sa počíta hash z CRL, a preto bola funkcia navrhnutá univerzálne. Funkcia LTES_set_other_hash dokáže spočítať hash z akejkoľvek ASN1 štruktúry. Funkcia príjme ako parameter typ hashovacieho algoritmu, ASN1 štruktúru

a metódu na kódovanie štruktúry do formátu DER. Hash hodnota štruktúry je potom počítaná z kódu štruktúry v DER formáte pomocou funkcie ASN1_digest.

Rozhranie knižnice LTES poskytuje funkciu LTES_add_otherSignCert, ktorá vytvorí atribút othetSignCert a pridá ho medzi podpisované atribúty. Funkncia príjma ako parametre štruktúru SignerInfo, podpisovaný a certifikát a typ hashovacieho algoritmu, ktorý sa použije na spočítanie otlačku z certifikátu. Vytvorený atribút otherSignCert bude mať vždy prítomné pole IssuerAndSerial a hash uloží do štruktúry OtherHashAlgAndValue.

5.9 Funkcia na výpočet hash hodnôt pre vydanie časových razítok

Časové razítka sa používajú na dôkaz existencie dát v určitom čase v minulosti. Dáta sú v časovom razítku identifikované na základe ich hash hodnoty, táto hodnota je v časovom razítku uložená v poli messageImprint v štrukúte TimeStampToken. Typ hashovacieho algoritmu môže byť ľubovolný, ale musí byť podporovaný autoritou pre výdaj časových razítok.

Štandard CAdES definuje štyri typy časových razítok, pridávaných ako nepodpisované atribúty

- signature-time-stamp v podpise CAdES-T
- CAdES-C-time-stamp v podpise CAdES-X Type1
- certificate-CRL-time-tamp v podpise CAdES-X Type 2
- archive-time-stamp v podpise CAdES-A

Hash sa spočíta z reťazca, ktorý vznikol spojením DER kódovaní určených polí (pre každý typ atribútu sú tieto polia iné). Pri atribútoch escTimestamp a certCRLTimestamp sa nepoužije DER kód celej položky, ale len DER kód poľa bez kódovania typu dát a dĺžky dát.

V OpenSSL sa dá jednoducho získať DER kódovanie ASN1 štruktúry pomocou príslušnej kódovacej funkcie (funkcie typu d2i), neexistuje však priamočiary postup na získanie DER kódovania štruktúry bez kódovania typu dát a dĺžky dát. V LTES preto vznikli funkcie, ktoré kódované informácie o type a dĺžke dát z DER kódu štruktúry odstránia.

LTES využíva funkciu OpenSSL ASN1_get_object. Funkcia ako parameter príjma referenciu na ukazateľ na pole znakov (DER kódovanie štruktúry) a po skončení nastaví túto referenciu na začiatok dátovej časti DER kódu. Z tohto sa potom dopočíta dĺžka vlastných dát, ktoré predstavujú DER kód štruktúry bez informácií o type dát a dĺžke dát.

5.10 Funkcie pri získanie ČR a vloženie ČR ako atribútu do podpisu

Štandard popisujúci časové razítka popisuje postup pre získanie časového razítka. Prvým krokom je vytvorenie žiadosti o časové razítko. Po zaslaní tejto žiadosti časovej autorite, časová autoria vydá časové razítko.

Tento postup je implementovaný aj v knižnici LTES. Knižnica LTES implementuje funkcie pre získanie žiadosti o všetky časové razítka, ktoré štandard CAdES definuje. Žiadosť o časové razítko je zapísaná do špeciálneho výstupného objektu BIO vo kódovaní DER. DER kódovanie je pre žiadosť o časové razítko vhodnejší, klientské aplikácie časových autorít totiž dokážu s týmto formátom pracovať.

LTES necháva spôsob získania časového razítka na užívateľovi. Časové razítko je totiž nutné získať od časovej autority a spôsob vydávania časových razítok sa u jednotlivých časových autorít líši. Každá klientská aplikácia časovej autority pre vydávanie časových razítok by však mala vedieť spracovať súbor so žiadosťou o časové razítko. Ak časová autorita poskytuje rozhranie aj v jazyku C, je možné funkcie knižnice LTES spojiť s voláním príslušného rozhrania a celý proces tvorby podpisu zautomatizovať.

LTES tiež implementuje funkcie, ktoré dostanú ako parameter časové razítko a toto časové razítko do nepodpisovaných atribútov pridajú. Tieto funkcie neoverujú platnosť časového razítko, overia len správnosť hash hodnoty vzhľadom k podpisu (či razítko zodpovedá podpisu).

5.11 Vytvorenie validačných referencií

Atribúty certificate-references a revocation-references sú nepodpisiované atribúty, ktoré obsahujú referencie na všetky certifikáty a revokačné informácie, potrebné pri overovaní podpisového certifikátu. Pre vytvorenie týchto atribútov je teda potrebné najskôr príslušné dáta získať a následne ich pridať do podpisu. Je dôležité pripomenúť, že získané validačné dáta by mali byť nespochybniteľné,a preto by CRL malo byť vydané až po čase vzniku podpisu.

Získanie validačných údajov

Pre získanie validačných dát bol využítý mechanizmus overovania podpisového certifikátu v OpenSSL. OpenSSL obsahuje funkciu X509_verify_cert, ktorá dostane ako parameter verifikačný kontext a zistí, či je zadaný certifikát platný alebo nie. Verifikačný kontext obsahuje sklad dôveryhodných a nedôveryhodných certifikátov, revokačných údajov a pod. Knižnica LTES rozširuje verifikačný kontext OpenSSL, aby bolo možné zachytiť chybové stavy pri overovaní a pozmeniť proces overovania certifikátu. Podrobne bol tento mechanizmus popísaný v kapitole o overovaní certifikátu v OpenSSL.

Po úspešnom procese overenie obsahuje verifikačný kontext informácie o certifikačnej ceste potrebnej pre overenie certifikátu, ako aj použitých revokačných údajoch. Proces získavania validačnčných údajov má v LTES nasledujúci postup:

1. inicializácia verifikačného kontextu, overenie certifikátu
2. z verifikačného kontextu sú získané informácie o potrebných certifikátoch
3. z verifikačného kontextu sú získané informácie o revokačných dokumentoch

Verifikačný kontext sa inicializuje volaním funkcie

```
int LTES_init_validation_info(LTES_VERIFY_CTX *ctx, X509 *cert,
X509_STORE *store,BIO *bio_err)
```

Táto funcie príjma ako parametre ukazateľ na alokovaný verifikačný kontext, podpisový certifikát (pre ktorý má získať validačné dáta), sklad certifikátov (potrebných pre overenie podpisového certifikátu) a výstup chybových správ. Ak funkcia vráti 1, prebehlo overenie podpisového certifikátu správne a verifikačný kontext je inicializovaný.

Pre získanie potrebných certifikátov slúží funkcia

```
STACK_OF(X509) *LTES_get_validation_X509(LTES_VERIFY_CTX *ctx)
```

Táto funkcia príjma ako parameter inicializovaný verifikačný kontext a vráti zoznam certifikátov, potrebných pre overenie popdisového certifikátu. Vráti ich v poradí, že dôveryhodná kotva je na konci zoznamu. Zoznam neobsahuje podpisový certifikát.

Pre získanie revokačných dát slúži funkcia

```
STACK_OF(X509_CRL) *LTES_get_validation_CRL(LTES_VERIFY_CTX
*ctx, X509 *cer)
```

Táto funkcia príjma ako parameter verifikačný kontext a certifikát. Funkcia vráti zoznam CRL, ktoré sú potrebné pre overenie platnosti certifikátu, ktorý je zadaný ako parameter. LTES definuje návratovú hodnotu ako typ zoznam certifikátov z dôvodu spracovania rozdielových CRL. Rozdielové CRL sa vzťahuje k normálnemu CRL a preto sú potrebné pre overnenie platnosti certifikátu dve CRL. Rozdielové CRL pri overovaní certfikátu zatiaľ OpenSSL nepodporuje a preto bude vždy vrátené jediné CRL v zozname. Prípadná pozdejšia podpora rozdielových CRL si ale nevyžiada zmenu rozhrania knižnice LTES.

Vytvorenie validačných referencií

Knižnica definuje funcie, ktoré vytvoria atribúty z validačnými referenciami:

`LTES_create_certificate_values(LTES_VERIFY_CTX *ctx)`

Funkcia na základe inicializovaného verifikačného kontextu vráti zoznam certifikátov potrebných pre overenie podpisového certifikátu

`LTES_create_revocation_values(LTES_VERIFY_CTX *ctx)`

Funkcia za základe inicializovaného verifikačného kontextu vráti zoznam revokačných údajov. Tieto údaje získa tak, že najprv získa zoznam použitých certifikátov a potom pre každý certifikát vola funkciu LTES_get_validation_CRL. Získané CRL vkladá do zoznamu, pričom nevkladá duplicitné CRL.

`LTES_create_revocation_refs(LTES_VERIFY_CTX *ctx, EVP_MD *md)`

Funkcia za základe inicializovaného verifikačného kontextu vráti zoznam revokačných referencií. Tieto získa tak, že najprv získa zoznam potrebných certifikátov a potom pre každý certifikát vola funkciu LTES_get_validation_CRL. Z každého CRL vytvorí referenciu a vloží ju do zoznamu referencií. Do referencií na CRL vkladá hash CRL, ktorý získa volaním funkcie LTES_set_other_hash, ale aj identifikáciu CRL podľa vydavateľa, dátumu vydania a čisla CRL.

`LTES_create_certificate_refs(LTES_VERIFY_CTX *ctx, EVP_MD *md)`

Funkcia na základe inicializovaného verifikačného kontextu vráti zoznam referencií na certifikáty. Referencie obsahujú hash certifikátu, získaný volaním fukcie LTES_set_other_hash, ale aj identifikáciu certifikátu podľa vydavateľa a sériového čisla certifikátu.

Volanie uvedených funkcií by mohlo byť pre užívateľa neprehľadné, a preto LTES implementuje funkciu

`int LTES_add_validation_attribute(PKCS7_SIGNER_INFO *si, X509 *signcert, X509_STORE *store, int flags).`

Funkcia príjme ako parameter štruktúru SingerInfo, podpisový certikát, skad certifikátov a špeciálny príznak. Funkcia podľa zadaného príznaku vloží do nepodpisovaných atribútov štruktúry SignerInfo príslušné atribúty. Ako špeciálny príznak je možné zadať:

- LTES_ADD_CERT_REFS - vloží atribút completeCertificateRefs
- LTES_ADD_REV_REFS - vloží atribút completeRevocationRefs

- LTES_ADD_CERT_VALS - vloží atribút certificateValues
- LTES_ADD_REV_VALS - vloží atribút revocationValue

Validačné referencie v časovom razítku

Štandard CAdES odporúča pridanie validačných referencií a dát aj do časového razítka. Zjednodušuje sa tým neskoršie dohľadanie potrebných certifkátov a CRL. LTES implementuje všetky potrebné funkcie pre pridanie týchto referencií a dát do časového razítka a riadková utilita LTES dokáže do existujúce časového razítka pridať tieto referencie.

Pri overovaní však tieto referencie nebudú zohľadnené. Modul časových razítok totiž zatiaľ nepodporuje overovanie časového razítka pomocou verifikačného kontextu. Nie je preto možné dostatočne upraviť priebeh overovania časového razítka tak, aby bolo možné validačné referencie zohľadniť.

5.12 Implementácia funkcií pre typ ASN1 Generalized Time

V časových razítkach je čas vydania uložený v položke Time v štruktúre TimeStampToken, ktorá tvorí v časovom razítku podpísanú správu. Čas vydania je uložený ako ASN1 typ GENERALIZED TIME, v OpenSSL mu zodpovedá typ ASN1_GENERALIZEDTIME. Tento typ podporuje štvorciferné číslo reprezentujúce rok, časové pásma a ľubovolnú presnosť zlomkov sekúnd. V časovom razítku je tiež uložená presnosť hodín časovej autority.

Pri implementácií knižnice LTES bolo potrebné získať čas typu time_t z typu ASN1_GENERALIZEDTIME. Čas je potrebné získať pri overovaní podpisu, kedy je nutné zistiť voči akému času má byť podpis overovaný, a tento čas je uložený práve v časovom razítku.

Knižnica LTES tiež pri zisťovaní najaktuálnejšieho časového razítka vyžadovala porovnanie časov v časových razítkach s ohľadom na presnosť uvedenú v časovom razítku.[4]

Knižnica LTES preto implemetuje dve funkcie na prácu s typom ASN1 GENERALIZED TIME:

```
int ASN1_GENERALIZED_TIME_get_ex(ASN1_GENERALIZEDTIME *a, time_t *tv_sec,
int *tv_usec)
```

Funkcia získa z času typu GERALIZED_TIME čas, ktorý rozdelí do dvoch zložiek. Do prvej zložky zapíše sekundovú časť v reprezentácií typom

[4]Nie je možné porovnať dva časy vzdialené desatinu sekundy, ak je presnosť oboch časov v rádoch desatín sekundy).

time_t, do druhej zložky zapíše počet mikrosekúnd.[5] Tieto zložky sú zapísané do ukazateľov, ktoré sú zadané ako druhý a tretí parameter funkcie.

Funkcia zohľadnuje časové zóny a presnosť typu GENERALIZED_TIME až na 6 desatinných miest. Vzhľadom na predlžovanie platnosti podpisu v niekoľkoročných intervaloch je poskytovaná presnosť dostatočná.

Spôsob implementácie funcie je pomerne zaujímavý a preto tu bude stručne popísaný. Funkcia najprv parsuje dátumovú a časovú zložku štruktúry. Z údajov o rokoch, mesiacoch, dňoch, hodinách, minútach a sekundách vytvorí štruktúru `tm`. Túto štruktúru potom skonvertuje na typ time_t pomocou funkcie mktime, funkcia mktime však do prepočtu zahŕňa aj aktuálnu časovú zónu užívateľa. Hodnota time_t je preto upravená o hodnotu premennej `timezone`. Jedná sa o systémovú premennú a obsahuje počet sekúnd tvoriaci rozdiel medzi Greenwichským časom a aktuálnym časom užívateľa. Nakoniec upravím hodnou time_t podľa informácie o časovej zóne v štruktúre ASN1_GENERALIZEDTIME. Funkcia podporuje časové zóny len s presnosťou na hodiny, posun o časti hodiny totiž vo svete nie je častý. Počet mikrosekúnd je zo štruktúry GENERALIZED_TIME získaný tiež parsovním.

```
ASN1_GENERALIZEDTIME_cmp(ASN1_GENERALIZEDTIME *a,sASN1_GENERALIZEDTIME *b,
LTES_TIME_PRECISION *precision)
```

Funkcia porovná dva časy GENERALIZED_TIME, pričom zohľadní informácie o presnosti oboch časov. Funkcia konvertuje časy volaním funkcie int ASN1_GENERALIZEDTIME_cmp do porovnateľných typov (time_t), potom ich po zohľadnení presnosti porovná.

Presnosť je zadaná ako tretí parameter a tvorí súčet presností oboch časov, presnosť má sekundovú, milisekundovú a mikrosekundovú zložku. Časy nie sú porovnateľné, ak je ich rozdiel menší ako súčet nepresností jednotlivých časov.

5.13 Vytvorenie podpisu podľa štandardov CAdES

Knižnica LTES bude obsahovať funkcie pre pridanie všetkých typov atribútov, ktoré definuje štadard CAdES. Umožní tak vytvorenie všetkých typov podpisov (okrem typu EPES), ktoré definuje štandard.

Knižnica LTES však nebude implementovať funkcie pre vytvorenie konkrétneho typu podpisu. Zostáva tak na programátorovi, aby pomocou funkcií LTES požadovaný typ podpisu vytvoril. Súčasťou tejto práce však bude aj riadková utlitia, ktorá umožňuje vytváranie jednotlivých druhov podpisov.

[5] Rozdelenie na zložky funguje rovnako ako u funkcie gettimeofday, ktorá je súčasťou OS Unix/Linux.

Funkcia	Popis
LTES_sign	Funkcia vytvori základný druh podpisu BES
LTES_add_otherSignCert	Funkcia pridá do podpisu atribút typu Other Signing Certificate
LTES_create_revocation_refs	Funkcia pridá do podpisu revokačné referencie
LTES_create_certificate_refs	Funkcia pridá do podpisu kompletné referencie na certifikátz
LTES_get_certificate_values	Funkcia pridá do podpisu hodnoty referencovaných certifikátov
LTES_get_revocation_values	Funkcia pridá do podpisu hodnoty revokačných dokumentov
LTES_create_CAdES_T	Funkcia pridá do podpisu atribút signature-time-stamp
LTES_create_CAdES_T1	Funkcia pridá do podpisu atribút CAdES-C-timestmap
LTES_create_CAdES_T2	Funkcia pridá do podpisu atribút cert-crl-timestmap
LTES_create_CAdES_A	Funkcia pridá do podpisu atribút archive-time-stamp
LTES_create_CAdES_C	Funkcia pridá do podpisu referencie na validačné dáta
LTES_create_CAdES_Long	Funkcia pridá do podpisu hodnoty validačných dát

Tabuľka 5.4: Funkcie, ktoré pridávajú atribúty do podpisu

5.14 Overovanie podpisu podľa štandardov CAdES

OpenSSL implementuje mechanizmus na overovanie elektronického podpisu a tiež mechanizmus na overenie certifikátu k času v minulosti. Prídavaním časových razítkok sa zabezpečuje dôkaz existencie podpisu k dátumu v minulosti, pridávaním referencií na dokumenty a samotných validačných dokumentov (certifikáty a CRL) sa zabraňuje podvrhnutiu týchto dokumentov. Knižnica LTES musí preto nielen implementovať funkcie pre overenie jednotlivých druhov atribútov, ale aj mechanizmus overenia celého podpisu.

Elektronický podpis môže obsahovať viacero vlastných podpisov, ktorým zodpovedajú štruktúry SignerInfo. Algoritmus overovania podpisu neoveruje celý podpis, ale jednotlivo každú SignerInfo štruktúru. To umožňuje overovať aj viacnásobné podpisy.

Algoritmus overí integritu jednotlivých atribútov v štruktúre SignerInfo tak, ako ich definuje štandard CAdES. Za prepokladu, že sú všetky podpi-

sované a nepodpisované atribúty správne, prebieha overenie podpisu nasledovným spôsobom:

1. v prípade, že existujú, z podpisu sa postupne odstraňujú archívne časové razítka. Najnovšie časové razítko je nutné overiť k aktuálnemu času, každé ďalšie k času vydania predchádzajúceho časového razítka. Zvyšok podpisu (po odstránení všetkých archívnych časových razítok) sa overuje k času vydania najstaršieho časového razítka.

2. v prípade, že existuje, overí sa časové razítko signature-time-stamp. Toto razítko overujem k času vydania najstaršieho archívneho časového razítka, alebo k aktuálnemu času, ak sa v podpise archívne časové razítka nenachádzali.

3. z podpisu sa získajú referencie na podpisový certifikát, certifikáty v certifikačnej ceste a príslušné certifkáty a CRL.

4. zistím dobu platnosti validačných dokumentov, ktoré sú referencované. Ak v dobe platnosti dokumentov vznikol podpis, podpisový certifikát overujem k času z atribútu signature-time-stamp. Ak sú však validačné dokumenty platné až po dobe vzniku podpisu, overujem podpis k času platnosti validačných dát.

5. pri overovaní podpisového certifikátu sa použijú len referencované validačné dokumenty. Validačné dokumenty, ktoré sú súčasťou podpisu, budú pri overovaní použité len ako nedôveryhodné (dôveryhodná kotva sa získa z dôveryhodného zdroja) a certifikát bude overený k času vydania časového razítka signature-time-stamp alebo k času platnosti validačných dokumnetov, prípadne k aktuálnemu času, ak sa časové razítko v podpise nechádza.

Skutočná implementácia je zložitejšia ako popísaný postup. Implementovaný algoritmus overuje jednotlivé atribúty a zároveň udržuje minimálnu hodnotu času, ku ktorému jednotlivé atribúty dokázateľne existovali. Tento čas získava z časových razítok. Jednotlivé časti podpisu sú potom k tomtuo času overované. Presný algoritmus, ktorým knižnica overuje podpis je takýto:

1. overí štruktúru PKCS7, typ štruktúry a skontroluje, či je podpísaná správa zahrnutá v podpise

2. spočíta hash podpísanej správy

3. postupne v bodoch 4–13 overuje jednotlivé podpisy (štruktúry SignerInfo)

4. nastaví čas, ku ktorému overuje T = aktuálny čas

5. ak je v podpise archívne časové razítko, vyber najnovšie razíko (A), overí ho a odstráni ho z podpisu, čas T = čas vydania razítka A

6. ak existuje ďalšie časové razítko, pokračuj bodom 5,inak choď na bod 7
7. over časové razítka nad podpisom typu CAdES-C Type 1 alebo Type 2
8. ak existujú, získa z podpisu validačné referencie
9. ak existujú, získa z podpisu validačné dokumenty, porovná ich s referenciami a vyberie len tie, ktoré sú referencované
10. overí atribút signature-time-stamp k času T. Ak sa atribút podarí overiť, nastaví čas T= čas vydania časového razítka signature-time-stamp.
11. stanoví dobu platnosti všetkých validačných dokumentov.
12. stanoví čas, ku ktorému bude overený podpisový certifikát. Tento čas určí na zákade platnosti validačných dokumentov a času pridania signature-time-stamp.
13. overí podpisované atribúty podpisu typu BES - ESS Signing Certificate alebo Other Signing Certificate, ktoré referencujú podpisový certifikát. Overí referenciu voči podpisovanému certifikátu.
14. overí, či súhlasí hash podpísanej správy s hashom uvedeným v podpise, overí hash podpisovaných atribútov
15. overí platnosť podpisového certifikátu voči času T. Pri overovaní použije sklad certifikátov, z ktorého odstráni nereferencované certifikáty. Pri overovaní použije podpisovaného certifikátu použije len referencované certifikáty a CRL.

Ak v niektorom z krokov overovanie zlyhá, podpis je označený za neplatný. V opačnom prípade je podpis označený za platný.

5.15 Overovanie atribútov obsahujúcich časové razítka

Pri overení atribútu obsahujúceho časové razítko je potrebné overiť, či časové razítko zodpovedá elektronickému podpisu (vypočítaná hodnota hash podľa štandardu je rovnaká ako v časovom razítku), ale aj platnosť samotného časového razítka.

Knižnica LTES implementuje funkcie pre overenie atribútov s časovými razítkami uvedené v tabuľke 5.5.

Každá funkcia dostane ako parameter čas, voči ktorému má časové razítko overiť. V prípade, že overované razítko predlžuje platnosť podpisu, funkcia hodnotu parametra nastaví na čas, voči ktorému sa má overovať zvyšok podpisu. Tento čas získa volaním funkcie `ASN1_GENERALIZED_TIME_get_ex`.

Funkcia	Popis
LTES_verify_CAdES_T	Overuje razítko z podpisu CAdES-T
LTES_verify_CAdES_Type1	Overuje razítko z podpisu CAdES-X Type 1
LTES_verify_CAdES_Type2	Overuje razítko z podpisu CAdES-X Type 2
LTES_verify_CAdES_A	Overuje razítko z podpisu CAdES-Archive

Tabuľka 5.5: Tabuľka uvádza zoznam funkcií pre overenie časového razítka

Keďže časové razítka môžu byť k podpisu pridávané niekoľko rokov, od rôznych časových autorít a pod rôznymi politikami, nedá sa vytvoriť univerzálna funkcia pre overenie časového razítka. Knižnica LTES preto umožňuje implementovat vlastnú funkciu pre overenie časového . Táto funkcia je zadaná ako parameter funkciám na overovanie časových razítok. Funkcia má nasledovné rozrhranie:

```
int verifyTST_method(PKCS7 *,time_t, int, X509_STORE *)
```

Ako parametre funkcia príjma :

- časové razítko
- čas ku ktorému má byť časové razítko overené
- NID idnetifikáciu atribútu, v ktorom je časové razítko uložené - napr. ako archívne časové razítko, alebo časové razítko typu signature-timestamp
- sklad certifikátov potrebných pre overenie certifikátu. Tento sklad je zhodný so skladom, ktorý bol zadaný ako parameter funkcií pre overenie celého atribútu.

Ak je časové razítko overené, funkcia vráti návratový kód 1, inak 0.

Užívateľ môže vytvorit vlastnú implementáciu funkcie pre overenie časového razítka a tak zabezpečiť napríklad akceptáciu razítok od rôznych časových autorít, alebo časové razítka vydané pod rozličnými politikami. Utilita LTES implementuje vzorovú funkciu pre overovanie časových razítok, ktorá pracuje interaktívne a kladie pri overovaní časového razítka užívateľovi dotazy.

5.16 Overenie archívnych časových razítok

Archívne časové razítka predlžujú platnosť celého podpisu a to tak, že dokazujú existenciu dôležitých častí podpisu k dátumu v minulosti. Hash pre archívne časové razítko sa spočítava aj z nepodpisovaných atribútov podpisu a teda aj zo starších archívnych časových razítok. Pri overovaní podpisu

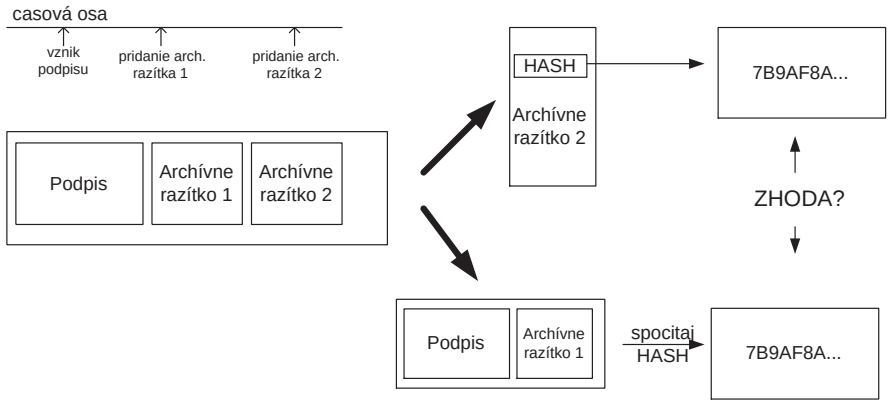
typu CAdES-A sa preto musí najnovšie časové razítko z nepodpisovaných atribútov odstrániť a až z takto upraveného podpisu je možné spočítať hash pre overenie odstráneného archívneho časového razítka.

Spôsob odstraňovania najnovšieho časového razítka je zaujímavý. Keďže nepodpisované atribúty tvoria štruktúru SET (teda množinu) a nie sekvenciu, nie je možné podľa poradia atribútov v zozname určiť poradie pridávania atribútov. Algoritmus nájdenia najnovšieho časového razítka preto prechádza všetky archívne časové razítka v nepodpisovaných atribútoch, každé razítko dekóduje a porovná čas vydania časového razítka so zatiaľ nájdeným maximánym časom. Po prehľadaní všetkých archívnych časových razítok vráti razítko s maximálnym časom vydania.[6]

Funkcia, ktorej volaním sa odstráni z podpisu najnovšie archívne časové razítko má syntax:

```
PKCS7 *LTES_remove_newes_ArchiveToken(PKCS7 *a,
    STACK_OF(X509_ATTRIBUTE) *sk, int remove_timestamp)
```

Funkcia ako parameter príjma zoznam nepodpisovaných atribútov, štruktúru PKCS7 a príznak, či má byť najnovšie archívne časové razítko odstránené. Funkcia vráti ukazateľ na najnovšie archívne časové razítko. Obrázok 5.1 ilustruje overovania archívneho časového razítka.



Obrázok 5.1: Ilustrácia overovania archívneho časového razítka

5.17 Overovanie podpisu typu BES

Knižnica OpenSSL implementuje mechanizmus pre overovanie elektronického podpisu podľa štandardu CMS. Pri overovaní podpisu typu BES je

[6]Pre porovnávanie časov vydania musela byť v LTES implementovaná funkcia na porovnávanie ASN1 typu GENERALIZED TIME.

nutné skontrolovať, či je podpísaná správa súčasťou podpisu a či sa medzi podpisovanými atribútmi nachádza atribút ESS Signing Certificate alebo Other Signing Certificate.

Aj keď knižnica LTES podporuje iba pridávanie atribútu typu Other Signing Certificate, dokáže overiť oba podpisované atribúty. Pri overovaní oboch atribútov sa najskôr porovná pole IssuerAndSerial atribútu s vydavateľom a poradovým číslom certifikátu, v prípade, že sú rozdielne, overenie atribútu sa nepodarí.

Na základe typu atribútu sa získa hodnota hash otlačku certifikátu a typ hashovacieho algoritmu. Na záver sa použitým typom hash algoritmu spočíta hodnota hash otlačku certifikátu a porovná sa s hodnotou, uvedenou v atribúte. Ak sa hodnoty otlačkov zhodujú, atribút je platný.

5.18 Úprava verifikačného kontextu

Pri overovaní podpisového certifikátu je možné použiť len tie certifikáty a CRL, ktoré sú referencované v podpise. Bežná funkčnosť OpenSSL to však neumožňuje, sklad certifikátov dohľadáva certifikáty až pri požiadavke na ich vyhľadanie a preto nie je možné nereferencované certifikáty a CRL pred začatím overovania zo skladu certifikátov odstrániť. Je preto nutné zmeniť priebeh overovania. Verifikačný kontext definovaný OpenSSL obsahuje ukazatele na funkcie, ktoré sa volajú pri požiadavke na dohľadanie certifikátu alebo CRL. Knižnica LTES upravuje verifikačný kontext a vkladá ukazatele na ňou definované funkcie pre získanie certifikátu a CRL, pričom pôvodné ukazatele na funkcie zapíše do rozšírenej štruktúry verifikačného kontextu. Verifikačnú kontext tiež rozšíri o ukazatele na validačné referencie.

Nahradením ukazatelov funkcii pre získanie certifikátu a CRL knižnica LTES upraví správanie týchto funkcií pri overení podpisového certifikátu. Funkcie implementované LTES zavolajú pôvodnú funkciu pre získanie certifikátu alebo CRL (ukazatele na pôvodné funkcie sú uložené v rozšírenom verifikačnom kontexte) a ak sa podarí nájsť hľadaný dokument, tento dokument ešte porovnajú s referenciami. Ak sa hľadaný dokument medzi referencami nenachádza, funcia oznámi, že sa hľadanie nepodarilo.

Príklad 5.12 ukazuje implementáciu rozšireného verifikačneho kontextu, spôsob uloženia pôvodných ukazateľov a nové funkcie pre vyhľadávanie certifikátu a CRL.

5.19 Overenie podpisového certifikátu k času

Pri overovaní podpisového certifikátu k dokázateľnému času existencie podpisu (uvedeného v atribúte signature-time-stamp) vzniká problém. Ak sa totiž pri vytváraní podpisu užívateľ rozhodol počkať s pridaním validačných referencií, aby boli referencované dokumenty nespochybniteľné (užívateľ zohľadnil grace period), validačné referencie boli do podpisu pridané neskôr

```
/* Pretazena funkcia na ziskanie certifikatu
* z ulozista. Vola vstavanu funkciu OpenSSL (get_issuer_original),
* ziskany certifikat este porovna s referenciami z podpisu
*/
int get_issuer(X509 **issuer, X509_STORE_CTX *ctx, X509 *x)
{
  int ok;
  int i,found;
  LTES_VERIFY_CTX *ltes_ctx = (LTES_VERIFY_CTX*)ctx;

  // pomocou povodnej funkcie ziska certifikat z ulozista
  ok = ((LTES_VERIFY_CTX*)ctx)->get_issuer_original(issuer,ctx,x);

  if( ltes_ctx->certRefs != NULL && ok > 0 )
  {
    found = 0;

    // prehladava referencie
    for(i=0;i<sk_LTES_OTHER_CERT_ID_num(ltes_ctx->certRefs);i++) {
      LTES_OTHER_CERT_ID *otherCert = sk_LTES_OTHER_CERT_ID_value(ltes_ctx->certRefs,i);
      if( LTES_verify_OTHER_CERT_ID(otherCert,*issuer) )
      found = 1;
    }

    ok = found;
  }

  return ok;
}

int LTES_init_verify_CTX(LTES_VERIFY_CTX *ctx,X509_STORE *store,STACK_OF(X509) *untrusted,
    time_t *t,LTES_COMPLETE_CERTIFICATE_REFS *certRefs,
    LTES_COMPLETE_REVOCATION_REFS *revRefs,int flags)
{
  // vytvor defaultny verifikacny kontext
  X509_STORE_CTX_init(&ctx->head,store,NULL,untrusted))

  // inicializuj konext
  X509_STORE_CTX_set_time( &ctx->head,0,*t);
  X509_STORE_CTX_set_flags(&ctx->head, flags );
  X509_STORE_CTX_set_purpose(&ctx->head, X509_PURPOSE_SMIME_SIGN);

  // nahrad defaultnu funkciu na ziskanie CRL novou funkciou
  ctx->get_crl_original = ctx->head.get_crl;
  ctx->head.get_crl = get_crl;

  // nahrad defaultnu funkciu na ziskanie certifikatu novou funkciou
  ctx->get_issuer_original = ctx->head.get_issuer;
  ctx->head.get_issuer = get_issuer;

  // do verifikacneho kontextu uloz validacne referencie
  ctx->certRefs = certRefs;
  ctx->revRefs = revRefs;

  return 0;
}
```

Príklad 5.12: Príklad ukazuje zmenu verifikačného kontextu a užívateľom definovanú funkciu na získanie certifikátu)

a v čase vzniku podpisu CAdES-T nemuseli byť referencované dokumenty platné. Týka sa to hlavne CRL, ktoré maju platnosť len niekoľko dní alebo hodín.

Riešením by bolo overovať certifikát k času vydania časového razítka signature-timestamp, a certifikát overovať voči CRL, ktoré v tom čase ešte neboli vydané. Za takýmto postupom stojí úvaha, že ak sa podpisový certifikát na takýchto CRL nenachádza (a teda nebol odvolaný ani v budúcnosti), bol v dobe overovanie taktiež platný. Overovanie certifikátov v OpenSSL však takúto funkčnosť nemá a preto by overenie voči CRL, ktoré ešte nie je platné zlyhalo - validačné dáta (CRL) by boli v čase overovania neplatné.

Riešením problému bez zmeny súčasnej implementácie overovania certifikátov v OpenSSL je posunutie času overovania podpisového certifikátu do doby, keď budú validačné dáta (a najmä CRL) platné. V tej dobe totiž tiež podpis určite existoval (dôkazom je časové razítko signature-time-stamp).

Čas platnosti validačných dokumentov sa určí tak, že sa spočíta maximum (*zac*) z časov začiatku platnosti a minimum (*kon*) z časov konca platnosti validačných dokumentov. V intervale (*zac,kon*) sú platné všetky validačné dokumenty, ak je tento inteval neprázdny. Algoritmus overovania podpisu takto zistí interval platnosti validačných dát uvedených v podpise. Ak bolo v tomto intervale vydané časové razítko signature-time-stam, užívateľ nezohľadnil grace period a podpisový certifikát bude overený k času vydania razítka. Ak však užívateľ zohľadnil grace-period, validačné dáta sú v čase vydania časového razítka neplatné a potom sa ako čas overovania podpisového certifikátu vezme ľubovolný čas z intevalu, v ktorom validačné dáta platili (napíklad stred tohoto intervalu).

Popísané riešenie problému však nefunguje v prípade, že validačné dáta a referencie nie sú súčasťou podpisu, to je prípad podpisu CAdES-T. Pred vlastným overovaním nie sú validačné dáta známe a preto nie je možné zistiť čas platnosti validačných dát. Tento problém by bolo možné riešiť tak, že by sa pred vlastným overením validačné dáta získali a z nich by sa získala doba platnosti. Takéto riešenia však nie je veľmi systematické, mohlo by spôsobiť zmätky pri vyhľadávaní správnych CRL. V prípade overovania podpisu typu CAdES-T bude preto algoritmus vyžadovať validačné dáta platné ku dňu vytvorenia popdpisu (resp. času vydania časového razítka).

Kapitola 6

Utilita LTES

Knižnica LTES umožňuje vytvárať všetky typy podpisov, ktoré definuje štandard CAdES. Implementuje funkcie, ktoré zjednodušujú prácu s atribútmi a umožňujú tvorbu a overovanie podpisu podľa štandardu CAdES. Pre potreby programátorov tak vznikla ucelená knižnica, ktorá umožňuje implementáciu rôznych riešení založených na štandarde CAdES.

Pre bežného užívateľa, ktorý si chce dlhodobo archivovať elektronický dokument, by však bola práca s funkciami a programovanie vlastného riešenia náročné. Súčasťou knižnice LTES je preto aj utilita, ktorá z príkazového riadku sprístupňuje najpoužívanejšie funkcie knižnice. Je vytvorená pre operačné systémy Windows aj Linux. Utilita umožňuje vytvárať a overovať všetky typy podpisov, ktoré popisuje štandard CAdES. Utilita bola implementovaná podľa vzoru utilít, ktoré sú súčasťou OpenSSL a preto by mala byť práca s ňou pre znalého užívateľa jednoduchá.

Podpisy typu CAdES-T,CAdES-Archive a pod. vyžadujú pridanie atribútu s časovým razítkom. Utlita je obecná a preto sa neviaže so žiadnou konkrétnou autoritou pre vydávanie časových razítok. Navrhnút obecný postup získania časového razítka nie je jednoduché. Každá časová autorita vydáva časové razítka iným spôsobom, líšia sa napríklad v použitom protokole alebo spôsobe autentizácie. Aby bolo vytváranie podpisov čo najobecnejšie, utlita LTES vytvorí len žiadosť o časové razítko vo formáte DER pre požadovaný typ podpisu. Užívateľ na základe žiadosti získa časové razíko, ktoré potom do podpisu pomocou utility LTES pridá [1]. Vytváranie podpisu s časovým razítkom má tak dva kroky, je ho však možné použiť s každou časovou autoritou.

Pri požiadavke na podpísanie správy, prípadne vytvorenie špecifického typu podpisu sa utilita riadi workflow, ktoré definuje štandard. Postup pri vytváraní podpisu je nasledovný:

1. vytvor podpis typu BES

[1] Spôsob získania časového razítka na základe žiadosti je ponechaný na užívateľovi, väčšina časových autorít poskytuje užívateľom klientské aplikácie na získanie časových razítok

2. vytvor žiadosť o časové razítko do atribútu signature-time-stamp
3. vlož atribút signature-time-stamp a vytvor podpis CAdES-T
4. vlož referencie na validačné dáta a vytvor podpis typu CAdES-C
5. vytvor žiadosť o časové razítko do jedného z podpisov CAdEC-C Type 1 resp. Type 2
6. vlož časové razítko a vytvor podpis typu CAdES-C Type 1 resp. Type 2
7. vlož validačné dáta a vytvor CAdES-Long
8. vytvor žiadosť o časové razítko do atribútu archive-time-stamp
9. vlož atribút signature-time-stamp a vytvor podpis CAdES-A

Pri vkladaní časového razítka sa očakáva, že umiestnenie časového razítko vo formáte PKCS7 zadá užívateľ. Užívateľ toto časové razítko obdrži od časovej autority na základe žiadosti o časové razítko, ktorú vytvorí utlita.

Pri požiadavke na vytvorenie podpisu s validačnými dátami alebo referenciami je potrebné utilite zadať umiestnenie certifikátov a CRL, ktoré sú potrebné pre overenie podpisu. Pri žiadosti o podpis typu CAdES-A dôjde vždy ku generovaní žiadosti resp. pridaniu archívneho časového razítka. Do podpisu je možné vkladať ľubovolný počet archívnych časových razítok. Čas pre predĺženie podpisu si musí určiť užívateľ, utilita LTES neumožňuje zistiť čas skončenia platnosti podpisu.

Pri požidavavke na overenie podpisu sa utilite nastaví úložište certifikátov a CRL, súbor s podpisom a prípadné prepínače. Utilita overí jednotlivé atribúty a oznámi, či je podpis platný alebo nie. Dodatok A obsahuje kompletnú užívateľskú dokumentáciu utility LTES.

6.1 Testovanie a používanie LTES

Testovanie každého produktu by malo obsahovať mimo iných aj funkčné a záťažové testy. Pri funkčných testoch bude vytvorené testovacie prostredie a v tomto prostredí bude dôkladne otestované vytváranie a overovanie podpisov podľa štandardu.

Keďže knižnica LTES vznikla na základe funkčných a nie výkonnostných požiadaviek, na záťažové testy nebol kladený dôraz. Čas vykonávania jednotlivých funkcií LTES pri testovaní funčnosti však dosahoval vždy hodnoty pod pol sekundy, čo postačuje pre vytváranie rádovo tisícov podpisov denne.

Veľký dôraz však bude kladený na testy spojené s použitím pamäte, jej alokovaním a uvoľnovaním, nakoľko použitie jazyka C zvyšuje pravdepodobnosť takýchto chýb.

Funkčné testy a testovacia certifikačná autorita

Testovanie všetkých funkcií LTES vyžaduje zložité testovacie prostredie. Za týmto účelom bola vytvorená testovacia certifikačná autorita, ktorá vydala všetky potrebné objekty pre testovanie. Jedná sa hlavne o podpisovací certifikát, koreňový certifikát, aktualizované zoznami CRL a autoritu pre vydávani časových razítok. Na vytvorenie dokumentov, potrebných pre otestovanie knižnice LTES bola použitá certifikačna autorita vytvorená pomocou OpenSSL.

Testovacie prostredie vytvorené pomocou OpenSSL bolo dostatočne zložité pre dôkladné otestovanie všetkých funkčností knižnice OpenSSL. Certifikačná autorita založená na OpenSSL splnila očakávania - práca s ňou bola jednoduchá a rýchla.

Testovanie použita pamäte

Pri programovaní v jazyku C hrozí veľké riziko chýb pri práci s pamäťou. Knižnica LTES bola preto dôkladne otestovaná pomocou software Valgrind.

Valgrind je produkt určený na odhaľovanie chýb pri práci s pamäťou. Je to open source produkt vydávaný pod licenciou GPL2, jeho použitie pri testovaní LTES preto nič nebránilo. Produkt má veľmi dobre referencie v programátorskej komunite a bol dokonca v roku 2008 ocenený ako najlepší nástroj pre vývoj OpenSource.

Testovanie pomocou nástroja Valgrind však nesplnilo očakávania. Štandardne Valgrind vypisuje zásobník volaných funkcií, kde došlo v memory-leak. V prípade testovania knižnice LTES však takúto informáciu nedokázal zobraziť a hľadanie memory-leak bolo preto veľmi pracné. Toto neočakávané správanie Valgrind mohlo byť spôsobené duplicitou knižnice OpenSSL, s ktorou si Valgrind nedokázal poradiť, alebo nedostatočnou znalosťou práce s nástrojom Valgrind.

Kapitola 7

Zhodnotenie splnenia cieľov práce a možnosti ďalšieho rozvoja

Ciele tejto diplomovej práce boli splnené. Bola vytvorená implementácia knižnice pre vytvorenie a overenie dlhodobo platného podpisu podľa štandardu CAdES. Knižnica LTES využíva knižnicu OpenSSL vo verzii 0.9.8, rozšírenú o modul OpenTSA na podporu časových razítok.

Je možné vytvárať a overovať všetky typy podpisu, ktoré definuje štandard CAdES (okrem štandardu EPES). Rozhranie knižnica LTES je dostatočne obecné a umožňuje vytvárať riešenia založené na štandarde CAdES. Knižnica LTES nepodporuje niektoré časti štandardu (rozdielové CRL, atribútové certifikátky), to je však dané existujúcou implementáciou knižnice OpenSSL. Ak sa OpenSSL bude ďalej rozvíjať, použiteľnosť knižnice sa bude zväčšovať.

Riadková utlita LTES poskytuje základné funčnosti knižnice a umožňuje vytvárať a overovať elektronický podpis. Súčasťou tejto práce je aj užívateľská príručka k tejto utilite. Nezanedbateľným prínosom tejto práce je zmapovanie existujúcej implementácie OpenSSL, dokumentácia samotnej OpenSSL je totiž veľmi slabá.

S súčasnej dobe, kedy neexistuje podobne detailná implementácia štandardu sa dá povedať, že je praktický prínos tejto práce značný. Zaujímavý je aj ďalší rozvoj tejto práce.

OpenSSL 0.9.9

V čase písanie tejto práce nie je knižnica OpenSSL vo verzií 0.9.9 vydaná oficiálne a nebol ani stanovený oficiálny termín vydania. Sú sice dostupné zdrojové kódy a aktuálny SNAPSHOT na CVS, obsahujú však chyby a nie sú pre implementáciu štandardov CAdES vhodné.

OpenSSL vo verzií 0.9.9 však pridáva mnoho nových funkčností a z pohľadu tejto práce sú zaujímavé najmä nasledovné:

- podpora časových razítok

- podpora štandardu CMS, ktorý dokáže spracovať aj podpisy zahrňujúce atribútové certifikáty
- zlepšený mechanizmus overovanie certifikátu
- zlepšený mechanizmus CRL, podpora obmedzených CRL

Zatiaľ sa zdá, že knižnica LTES bude s OpenSSL vo verzií 0.9.9 spätne kompatibilná. Určite by však bolo zaujímavé rozšíriť modul LTES aj o nové funkčnosti OpenSSL 0.9.9, je totiž pravdepodobné, že sa OpenSSL bude ďalej rozvíjať.

Jedným z cieľov tejto práce bolo získat informácie a prípadne sa pokúsiť o začlenenie knižnice LTES priamo do oficiálnej distribúcie OpenSSL. Podľa pokynov vydaných komunitou vývojárov knižnice OpenSSL, pridať novú funkčosť od OpenSSL je možné. Je nutné zaslať patch (rozdiel medzi novou a pôvodnou distribúciou) na e-mailovú adresu openssl-dev@openssl.org. Keďže pridané funkčnosti knižnice LTES sú značné, bolo pridanie do LTES do OpenSSL predbežne diskutované s vývojármi OpenSSL. Vývojári OpenSSL o knižnicu prejavili záujem. V ďalšej fáze bude preto vytvorený patch na verziu 0.9.8 a zaslaný tímu OpenSSL spolu s dokumentáciou.

Aplikácia pre vytváranie a overovanie podpisu

Ako ďalší rozvoj tejto práce by mohla byť implementácia aplikácie pre vytvváranie a overovanie dlhodobého elektronického podpisu. Existujúci nástroj v podobe riadkovej utlity sa pre bežného užívateľa nehodí. Mohla by preto vzniknúť grafická a platformovo nezávislá aplikácia, ktorej použitie by bolo intuitívne. Podobné grafické nástroje na vytvárenie dlhodobého elektronického podpisu už existujú, jedná sa však o komerčné nástroje a nepodporujú pridávania archívnych časových razítok.

Kapitola 8

Záver

Štandard CAdES predstavuje zaujímavé riešenie dlhodobej archivácie dokumentov. Používa mechanizmus predlžovania platnosti eletronického podpisu pomocou časových razítok a je platným odporúčaním Európskej komisie pre dlhodobé archivovanie dokumentov.

Táto práca si kládla sa úlohu implementovať štandard do open source produktu OpenSSL. Tento cieľ sa podarilo splniť, aj keď nie je štandard implementovaný dokonale. Súčasná implementácia knižnice umožňuje vytvárať elektronický podpis, ktorého platnosť je možné predlžovať. Takýto podpis je tiež možné overovať. V súčasnosti neexisuje iná Open Source alternatíva implementácie štandardu CAdES a ani komerčné implementácie nepokrývajú štandard CAdES v takej šírke, v akej ho pokrýva knižnica LTES.

Potencionálne využitie knižnice LTES v praxi je značné. Mohla by byť ideálnym riešením pri návrhu systému pre archiváciu dokumentov. Knižnicu OpenSSL a teda ani knižnicu LTES však nie je možné použiť v riešeniach vyžadajúcich certifikované produkty.

Zaujímavý rozvoj tejto práce predstavuje zahrnutie knižnice LTES priamo do oficiálnej distribúcie OpenSSL vo verzií 0.9.9, alebo implementácia grafického rozhrania pre jednoduchú tvorbu dlodobo archivovateľných podpisov pre bežného užívateľa.

8.1 Výhody a nevýhody archivovania dokumentov s využitím štandardu

Predlžovanie platnosti elektronického podpisu podľa štandardu CAdES je založené na "prerazítkovávaní" podpisu. Keď sa blíži koniec platnosti podpisu, k štruktúre elektronického podpisu sa pridá nové časové razítko a doba platnosti novej štruktúry bude určená platnosťou pridaného časového razítka. Dobu platnosti časového razítka určuje certifikát časového servera a táto platnosť je sa môže u jednotlivých časových autorít značne líšiť (v súčasnosti platný certifikát časovej autority Prvni certifikační autority (ICA) má dobu platnosti 5 rokov).

Pred vypršaním doby platnosti časového certifikátu je nutné všetky elektronické podpisy, ktorých platnosť chceme predĺžiť prerazítkovať. Tento proces určite nie je problémom, ak archivujeme niekoľko stoviek alebo tisíc podpisov, problém však nastáva u veľkých el. archívov s miliónmi dokumentov.

Proces prerazítkovania je nutné zohľadniť pri návrhu archívu, nakoľko záťaž spôsobena prerazítkovávaním môže byť značná a je dobré ju časovo naplánovať a prerazítkovávať s predstihom (nový certifikát časového servra je vydávaný v prípade ICA niekoľko mesiacov pred vypršaním doby platnosti staršieho certifikátu).

Ďalší problém spojený s prerazítkovávaním je finančný - cena časového razítka je je napr. v prípade ICA tvorená dohodou a môže sa nachádzat v pásme 0.2 – 2 CZK. Cena milióna časových razítok potom môže byť značná položka a je s ňou nutné v nákladoch na prevádzaku systém počítať.

Na základe horeuvedených skutočností je vhodné archivovať dokumenty pomocou štandardu CAdES len na omezenú dobu - niekoľko rokov, kde nie je nutné dokumenty prerazítkovať a stačí časové razítko pridané pri tvorbe podpisu. Pri požiadavke na dlhodobú archiváciu (viac ako 20 rokov) by bolo vhodné zvážiť aj iné spôsoby archivácie, ktorých nákupné náklady sú síce väčšie, prevádzkové náklady ich však vyvážia.

Dodatok A

Užívateľská dokumentácia pre utilitu LTES

Súčasťou knižnice LTES je riadková utilita, ktorá sprístupňuje najčastejšie používané funkčnosti knižnice LTES z príkazovej riadky. Vnikla podľa zvyklostí OpenSSL. Utilita LTES umožňuje vytvárať a overovať všetky druhy podpisov podľa štandardu CAdES. Utilitu je možné použiť pod operačnými systémami Windows XP,2000 a Linux resp. Unix.

A.1 Inštalácia

Windows

Pod operačným systémom Windows je utilita zostavená ako exe súbor. Je staticky zlikovaná s knižnicou LTES a dynamicky zlikovaná s knižnicou OpenSSL (s modulom OpenTSA). Utilita je na dodanom CD uložená v adresári LTES/Windows/app spolu so všetkými súbormi, potrebnými pre beh knižnice. Spustiteľný súbor má názov ltes.exe.

Pred vlastným spustením je nutné mať nainštalovaný software C++ Redistribution Pack, ktorý je súčasťou dodaného CD. Tento balík obsahuje knižnice potrebné pre beh samotnej OpenSSL ako ja utility LTES.

Linux

Pod operačným systémom Linux je knižnica zostavená ako spustiteľný súbor, pričom sa dynamicky linkuje s knižnicou LTES a OpenSSL (s podporou časových razítok). Pred zostavením utlitity je nutné najprv skompilovať samotnú knižnicu LTES. Zdrojové kódy knižnice sa nachádzajú v adresári LTES/Linux/src. Knižnica sa zostavuje postupnosťou príkazov make a make install. Knižnica sa nainštaluje do adresára /usr/local, umiesnenie je možné zmeniť v Makefile. Zdrojový kód utility je uložený na dodanom CD v adresár LTES/Linux/app. Tieto zdrojové kódy sa zostavia nástrojom make.

Spustitelný súbor má názov ltes. V súboroch Makefile knižnice a utility je možné upraviť cesty ku knižnici OpenSSL (makro OPENSSL_PATH).

A.2 Vstupy utility LTES

Digitálny podpis

Digitálny podpis je základný vstup utility LTES. Zadáva sa v prepínači –in <file>. Digitálny podpis sa zadáva ako vstup pri požiadavke na pridanie atribútov alebo overenie podpisu. Pri požiadavke na pridanie atribútov do podpisu sa nekontroluje integrita ani platnosť podpisu. Preto ak bol podpis pred pridaním atribútu neplatný, vytvorený podpis bude tiež neplatný. Podpis je možné zadať vo formáte DER alebo PEM, formát podpisu sa špecifiuje prepínačom –inform DER—PEM.

Certifikáty a CRL

Pre overenie platnosti podpisu, ale aj pre vytvorenie validačných referencií, je potrebné určiť, odkiaľ sa majú získať validačné dokumenty. Utilita LTES podporuje načítanie súboru s CRL alebo certifikátom, ale aj načítanie adresárov s CRL resp. certifikátmi. Pre načítanie CRL resp. certifikátu zo súboru sa použije prepínač –CRLfile <file> resp. –CAfile <file>, príslušný dokument má byť uložený vo formáte PEM. Pre načítanie celého adresára s CRL resp. certifikátmi sa použije prepínač –CRLpath <dir> resp. –CApath <dir>. Dokumenty musia byť uložené vo formáte PEM. Adresár s certifikátmi sa musí pred volaním utility spracovať volaním utility c_rehash.

Umiesnenie validačných dát je nutné zadať pri požiadavke na overenie podpisu a tiež pri požiadavke na pridanie atribútu s validačnými referenciami resp. dokumentami.

Podpisový certifikát a súkromný kľúč

Pri podpísaní správy je potrebné zadať ulitite podpisový certifikát spolu so súkromným kľúčom. Utlita LTES podporuje dva spôsoby zadania súkromného kľúča a podpisového certifikátu.

Súkromný kľúč a certifikát je možné zadať vo formáte PKCS12. V tomto type súboru sa dajú uložiť súkromný kľúč spolu s certifikátom. Súbor PKCS12 sa zadá v prepínači –signkey <file>, heslo k súboru PKCS12 sa zadáva prepínačom –passin <password>. Súkromný kľúč a podpisový certifikát je však možné načítať aj osobitne - z jedného súboru sa načíta súkormný kľúč, z ďalšieho potom podpisový certifikát. V takomto prípade je však nutné určiť typ súkromého kľúča ako PEM pomocou prepínača –keyform PEM. Súbor so súkromným kľúčom sa potom určí prepínačom –signkey <file> a súbor s certifikátom prepínačom –certfile <file>. Heslo k súkromnému kľúču sa zadá v prepínači –passin <password>.

Poznámka: Z dvojice súkromný kľúč, certifikát sa dá vytvoriť PKCS12 súbor príkazom

```
openssl pkcs12 -export -in <certifikat> -inkey <suk_kluc> \
               -out <output.pfx>
```

Pri vytváraní dokumentu PKCS12 bude od užívateľa vyžiadané heslo, ktorým bude súkromný kľúč chránený.

Časové razítko

Pri požiadavke na pridanie časového razítka ako nepodpisovaného atribútu do podpisu sa časové razítko zadáva prepínačom –TSTin <file>. Časové razítko musí byť vo formáte DER a musí ho tvoriť štruktúra PKCS7 (a nie Timestamp Response). Pri vkladaní razítka do podpisu sa kontroluje, či razítko obsahuje správny hash, nekontroluje sa však platnosť razítka. Overenie platnosti razítka musí vykonať užívateľ (môže použiť napríklad funkciu overenia časového razítka v OpenSSL).

Časové razítko sa obrdží na základe žiasosti o časové razítko, ktorú generuje utlita LTES. Ak chce užívateľ pridať napr. razítko signature-time-stamp, vygeneruje najprv z podpisu žiadosť o časové razítko, na základe žiadost obrdží od časovej autority časové razítko a nakoniec pomocou utility LTES pridá razítko ako atribút signature-time-stamp do podpisu. Pridávania atribútov s časovými razítkami teda prebieha v dvoch krokoch, tento postup je však univerzálny a nie je viazaný na žiadnu časovú autoritu. (Priame získanie časového razítka by si vyžadovalo prispôsobenie utility pre konkrétnu časovú autoritu.)

Poznámka: Časové razítko sa dá vygenerovať zo žiadosti o časové razítko aj nástrojom OpenSSL s modulom OpenTSA. Ak je tento modul správne nakonfigurovaný, časové razítko sa vydá na základe žiadosti príkazom:

```
openssl ts -reply -queryfile <request.tsq> \
        -out <timestamp.tst> -token_out
```

A.3 Výstupy utility LTES

Žiadosť o časové razítko

Utilita LTES dokáže spracovať digitálny podpis a vygenerovať žiadosť o časové razítko, ktoré má byť pridané do podpisu ako nepodpisovaný atribút. Súbor, do ktorého bude zapísaná žiadosť o časové razítko, sa zadáva pomocou parametra -TSQout. Žiadosť o časové razítko je vygenerovaná vo

formáte DER. Utilita LTES umožňuje špecifikovať politiku, pod ktorou má byť vydané časové razítko, pomocou prepínača –policy <policy_id>. Identifikátor politiky je v tvare OID, napr. 120.23.344.3.4. Ak tento prepínač nie je použitý, žiadosť o časové razítko nebude obsahovať identifikátor politiky.

Digitálny podpis

Výstupom knižnice LTES je digitálny podpis. V prípade overenia digitálneho podpisu alebo vygenerovania časového razítka, je digitálny podpis na výstupe zhodný s podpisom na vstupe (zadaným pomocou parametra –in). V prípade pridávania atribútov bude na výstupe nový podpis s pridanými atribútmi. Digitálny podpis sa typicky vypíše na štadardný výstup, pomocou prepínača –out <file> je možne špecifikovať súbor, kam sa má podpis uložiť. Formát podpisu je možné špecifikovať prepínačom –outform <PEM—DER>. Výstup sa dá potlačiť prepínačom –noout.

Podpísaná správa

Digitálny podpis podľa štandardu CAdES vždy obsahuje aj podpísanú správu. Túto správu je možné zapísať do súboru prepínačom –outdata <file>. Tento prepínač je možné použiť v kombinácií s ľubovolným iným prepínačom, správu je teda možné uložiť pri požiadavke na pridanie atribútu do podpisu ale aj pri overovaní podpisu.

A.4 Použitie utility LTES

Utilitu je možné použiť na vytvorenie a overenie podpisu podľa štandardu CAdES. Vytvorenie a overenie podpisu budú popísané samostatne.

Utility v normálnom režime nevypisuje mnoho sprievodných správ, použitím prepínača -verbose sa zväčší počet správ.

Vytvorenie podpisu

Pri vytváraní podpisu sa postupuje v jednotlivých krokoch tak, ako ich popisuje štandard CAdES. Pre vytvorenie podpisu sa utilite CAdES zadá parameter –sign. Požadovaný typ podpisu sa udáva v parameteri –type. Tabuľka A.1 uvádza hodnoty prepínača type pre jednotlivé typy podpisov.

Poznámka: Aj keď nie sú podpisy typu CAdES-Archive Type 1 resp. Type 2 súčasťou štandardu CAdES, utilita LTES ich zavádza. Súbor typu CAdES-Archive vyžaduje časové razítko nad validačnými referenciami a práve Type 1 resp. Type 2 udáva, o ktoré razítko sa má jednať. Pri požiadavke na vytvorenie archívneho podpisu nad typom CAdES-X Long Type 1 resp. Type 2 sa preto použije prepínač A1 resp. A2.

Typ podpisu	Hodnota prepínača	Základ podpisu
CAdES-BES	BES	-
CAdES-T	T	CAdES-BES
CAdES-C	C	CAdES-T
CAdES-X Type 1	C1	CAdES-C
CAdES-X Type 2	C2	CAdES-C
CAdES-X Long	L	CAdES-C
CAdES-X Long Type 1	L1	CAdES-X Long
CAdES-X Long Type 2	L2	CAdES-X Long
CAdES-Archive Type 1	A1	CAdES-X Long Type 1
CAdES-Archive Type 2	A2	CAdES-X Long Type 2

Tabuľka A.1: Tabuľka udáva hodnoty prepínača –type pre jednotlivé druhy podpisov

VYTVORENIE PODPISU BES

Na základe podpisovaných dát, súkromného kľúča a podpisového certifikátu sa vytvorí podpis typu BES. Súkromný kľúč a podpisový certifikát sú uložené buď v súbore typu PKCS12, alebo v samostatných súboroch vo formáte PEM. V prípade, že je súkromný kľúč chránený heslom, heslo sa zadá sa ako parameter utility. Nasledujúce príklady ukazujú použitie utility:

```
ltes.exe -sign -type BES -out sk.sgn -indata sk \
         -keyfile user.pfx -passin test -outform DER
```

Bude vytvorený podpisu typu BES. Podísaná správa sa načíta zo súboru sk,súkromný kľúč a certifikát sa načítajú zo súboru user.pfx, chráneného heslo test. Podpis bude uložený vo forme DER do súboru sk.sgn.

```
ltes.exe -sign -type BES -out sk.sgn -indata sk \
         -certfile user.pem -keyfile user_key.pem
```

Bude vytvorený podpisu typu BES. Podísaná správa sa načíta zo súboru sk,súkromný kľúč sa načíta zo súboru user_key.pem a nie je chránený heslom. Podpisový certifikát sa načíta zo súboru user.pem. Podpis bude uložený vo forme PEM do súboru sk.sgn.

VYTVORENIE PODPISU CAdES-T

Vytvorenie podpisu s časovým razítkom prebieha v dvoch krokoch. V prvom kroku sa na základe podpisu typu BES vygeneruje žiadosť o časové razítko atribútu signature-time-stamp. Na základe tejto žiadosti užívateľ získa od časovej autority časové razítko. V druhom kroku sa potom toto časové razítko do podpisu typu BES pridá. Vzniká tak podpis CAdES-T. Nasledujúce príklady ukazujú použitie utility:

```
ltes.exe -sign -in sk.sgn  -TSQout ts-signatureTST.tsq \
         -type T -noout -policy 1.2.3.4
```

Pre súbor s podpisom sk.sgn utilita vytvorí žiadosť o časové razítko vo formáte DER (nie je možné zmeniť) a uloží ju do súboru ts-signatureTST.tsq. Pre túto žiadosť získa užívateľ časové razítko.

```
ltes.exe -sign -in sk.sgn -TSTin ts-signatureTST.tst \
         -type T -out sk.CAdES-T.sgn
```

Do súboru s podpisom sk.sgn vloží utilita LTES časové razítko zo súboru ts-signatureTST.tst. Výsledný podpis zapíše do súboru sk.CAdES-T.sgn vo formáte PEM.

VYTVORENIE PODPISU TYPU CAdES-C

Podpis typu CAdES-C sa vytvára z podpisu typu CAdES-T. Validačné údaje potrebné pre overenie certifikátu sa utilite zadajú pomocou parametrov CRLpath,CApath, CRLfile a CAfile. Ako parameter -type sa zadá znak C. Nesledujúci príklad uvádza vytvorenie podpisu typu C:

```
ltes.exe -sign -in sk.CAdES-T.sgn -type C -out sk.CAdES-C.sgn \
         -CRLpath ./crl -CApath ./certs
```

Bude vytvorený podpis typu CAdES-C, ktorý bude uložený do súboru sk.CAdES-C.sgn. Pri tvorení validačných referencií sa použijú CRL umiestnené v adresári crl a certifikáty umiestnené v adresári certs.

VYTVORENIE PODPISU TYPU CAdES-X Long

Podpis typu CAdES-X Long sa vytvára z podpisu typu CAdES-C. Validačné údaje potrebné pre overenie certifikátu sa utilite zadajú pomocou parametrov CRLpath,CApath, CRLfile a CAfile. Ako parameter -type sa zadá znak L. Nesledujúci príklad uvádza vytvorenie podpisu typu Long:

```
ltes.exe -sign -in sk.CAdES-C.sgn -type L -out sk.CAdES-L.sgn \
         -CRLpath ./crl -CApath ./certs
```

Bude vytvorený podpis typu CAdES-X Long, ktorý bude uložený do súboru sk.CAdES-L.sgn. Pri tvorení validačných dát sa použijú CRL umiestnené v adresári crl a certifikáty umiestnené v adresári certs.

VYTVORENIE PODPISU TYPU CAdES-X Type 1 resp. Type 2

Podpis typu CAdES-X Type 1 resp. Type 2 sa vytvára z podpisu typu CAdES-C. V prvom kroku sa na základe podpisu typu CAdES-C vygeneruje žiadosť o časové razítko atribútu escTimestamp resp. certCRLTimestamp. Na základe tejto žiadosti užívateľ získa od časovej autority časové razítko. V druhom kroku sa potom toto časové razítko do podpisu typu CAdES-C pridá. Vzniká tak podpis CAdES-X Type 1 resp. Type 2. Type 1 alebo Type 2 sa vytvorí na základe prepínača -type "typ". Ak je zadaný typ C1, vytvorí sa podpis typu CAdES-X Type 1. Ak je zadaný typ C2, vytvorí sa podpis typu CAdES-X Type 2. Nasledujúce príklady ukazujú použitie utility:

```
ltes.exe -sign -in sk.CAdES-C.sgn -type C1 \
         -TSQout sk-C-T1.tsq -policy 1.2.3.4 -noout
```

Príkazom bude vygenerovaná žiadosť o čas. razítko CAdES-C-timestamp, žiadosť bude uložená do súboru sk-C-T1.tsq vo formáte DER. Časové razítko má byť vydané pod politikou 1.2.3.4. Digitálny podpis na výstupe bude potlačený prepínačom noout.

```
ltes.exe -sign -in sk.CAdES-C.sgn -type C1 -TSTin sk-C-T1.tsr \
         -out sk.CAdES-X_T1.sgn
```

Príkazom bude vygenerovaný podpis typu CAdES-X Type 1 a bude uložený do súboru sk.CAdES-X_T1.sgn. Časové razítko bude získané so súboru sk-C-T1.sgn, kde je uložené vo formáte DER.

VYTVORENIE PODPISU TYPU CAdES-X Long Type 1 resp. Type 2

Podpis typu CAdES-X Long Type 1 resp. Type 2 sa vytvára z podpisu typu CAdES-X Long. V prvom kroku sa na základe podpisu typu CAdES-X Long vygeneruje žiadosť o časové razítko atribútu escTimestamp resp. certCRL-Timestamp. Na základe tejto žiadosti užívateľ získa od časovej autority časové razítko. V druhom kroku sa potom toto časové razítko do podpisu typu CAdES-X Long pridá. Vzniká tak podpis CAdES-X Long Type 1 resp. Type 2. Type 1 alebo Type 2 sa vytvorí na základe prepínača -type "typ". Ak je zadaný typ L1, vytvorí sa podpis typu CAdES-X Long Type 1. Ak je zadaný typ L2, vytvorí sa podpis typu CAdES-X Long Type 2. Nasledujúce príklady ukazujú použitie utility:

```
ltes.exe -sign -in sk.CAdES-L.sgn -type L1 -TSQout sk-L-T1.tsq \
         -policy 1.2.3.4 -noout
```

Príkazom bude vygenerovaná žiadosť o čas. razítko CAdES-C-timestamp, žiadosť bude uložená do súboru sk-L-T1.tsq vo formáte DER. Časové razítko má byť vydané pod politikou 1.2.3.4. Digitálny podpis na výstupe bude potlačený prepínačom noout.

```
ltes.exe -sign -in sk.CAdES-L.sgn -type L1 -TSTin sk-L-T1.tsr \
         -out sk.CAdES-L_T1.sgn
```

Príkazom bude vygenerovaný podpis typu CAdES-X Long Type 1 a bude uložený do súboru sk.CAdES-L_T1.sgn. Časové razítko bude získané so súboru sk-L-T1.sgn, kde je uložené vo formáte DER.

VYTVORENIE PODPISU TYPU CAdES-Archive

Podpis typu CAdES-Archive sa vytvára z podpisu typu CAdES-X Long Type 1 alebo Type 2. V prvom kroku sa na základe podpisu typu CAdES-X Long Type 1 resp. Type 2 vygeneruje žiadosť o časové razítko atribútu archiveTimetamp. Na základe tejto žiadosti užívateľ získa od časovej autority časové razítko. V druhom kroku sa potom toto časové razítko do podpisu typu CAdES-X Long pridá. Vzniká tak podpis CAdES-Archive. Podpis typu archive môže mať dva tvary - v závislosti od toho či vznikol z podpisu CAdES-X Long Type 1 alebo Type 2.

Nasledujúce príklady ukazujú použitie utility:

```
ltes.exe -sign -in sk.CAdES-L_T1.sgn -type A1 \
         -TSQout sk-A-T1.tsq -policy 1.2.3.4 -noout
```

Príkazom bude vygenerovaná žiadosť o čas. razítko CAdES-C-timestamp, žiadosť bude uložená do súboru sk-A-T1.tsq vo formáte DER. Časové razítko má byť vydané pod politikou 1.2.3.4. Digitálny podpis na výstupe bude potlačený prepínačom noout.

```
ltes.exe -sign -in sk.CAdES-L_T1.sgn -type A1 \
         -TSTin sk-A-T1.tsr  -out sk.CAdES-A_T1.sgn
```

Príkazom bude pridané archívne časové razítko do podpisu zo súboru sk.CAdES-L_T1.sgn. Časové razítko bude získané so súboru sk-A-T1.sgn, kde je uložené vo formáte DER. Výsledný podpis bude uložený do súboru sk.CAdES-A_T1.sgn.

Overenie podpisu

Pre overenie podpisu sa použije prepínač –verify. Súbor s podpisom sa zadá pomocou parametra -in. Formát súboru s podpisom určuje prepínač -inform, je možné zvoliť medzi kódovaním DER a PEM. Pre potreby overovania certifikovátov z podpisu je potrebné určiť úložište certifikátov a CRL. Umiestnenia validačných dát sa nastavujú pomocou parametrov –CRLpath, –CApath, –CRLfile a –CAfile. Na výstup sa zobrazí priebeh overovania časových razítok a podpisového certifikátu. Na konci výstupu je informácia, či je podpis platný, spolu s informáciou o type podpisu, prípadne informácia, že overovanie zlyhalo. Podporované typy podpisov sú CAdES-BES, CAdES-T,CAdES-C,CAdES-X Type 1, CAdES-X Type 2, CAdES-X Long Type 1, CAdES-X Long Type2 a CAdES-A (Type 1 resp. Type 2). Príklad ukazuje overenie podpisu:

```
ltes.exe -verify -in sk.CAdES-L.sgn -outdata msg..txt \
         -CRLpath ./crl -CApath ./certs -noout
```

Príkaz overí podpis uložený v súbore sk.CAdES-L.sgn. Certifikáty a CRL za získajú z uvedených adresárov. Popri procese verifikácie bude uložená podpísaná správa do súboru msg.txt.

Použitie špeciálych prepínačov pri overovaní

Pri overovaní je možné použiť špeciálne prepínače, ktoré upravia priebeh overovania elektronického podpisu.

Prepínačom *–nocrl* sa špecifikuje, že pri overovaní podpisu nemajú byť brané do úvahy CRL. Tento prepínač sa hodí vtedy, keď nie je pochybnosť o tom, že certifikáty neboli odvolané.

Prepínačom *–trust_long* sa špecifikuje, že certifikáty uložené v podpise ako validačné dáta môžu byť podkladané za dôveryhodné. V opačnom prípade sú použíte len ako nedôveryhodné a utilita musí získať dôveryhodný koreňový certifikát z iného zdroja. Pri používaní tohoto atribútu je však nutné postupovať opatrne, certifikátom v podpise sa totiž obecne nedá dôverovať.

A.5 Úplný zoznam prepínačov

```
-inform arg     - vstupný formát podpisu (PEM alebo DER)
-outform arg    - výstupný formát podpisu (PEM alebo DER)
-indata arg     - súbor arg s podpisovanou správou
-outdata arg    - do súboru arg zapíše podpísanú správu
```

```
-signkey arg    - súbor so súkromným kľúčom
-signcert arg   - súbor s podpisovým certifikátom
-keyform arg    - formát kľúča (PEM and PFX)
-in arg         - vstupný súbor s podpisom, predovelne stdin
-out arg        - výstupný súbor s podpisom, predovlene stdou
-passin arg     - heslo k súboru so súkromným kľúčom
-TSTin          - súbor s časovým razítkom
-TSQout         - do súboru arg bude zapísaná žiadosť o časové razítko
-CApath arg     - úložište dôveryhodnýhch certifikátov vo formáte PEM
-CAfile arg     - súbor s certifikátmi
-CRLpath arg    - úložište CRl vo formáte PEM
-CRLfile arg    - súbor s CRL
-type arg       - typ podpisu, ktorý má byť vytvorený
-sign           - požiadavok na vytvorenie podpisu
-verify         - požiadavok na overenie podpisu
-policy         - špecifiuje politiku časového razítka
-noout          - nevypisuj podpis na výstup
-verbose        - mnoho sprievodných správ
-nocrl          - nepoužívaj CRL pri overovaní podpisu
-trust_long     - certifikáty v atribúte cert-values sú dôveryhodné
```

Dodatok B

Obsah priloženého CD

Súčasťou práce je CD obsahujúce knižnicu LTES, testovacie certifikáty a CRL, ukážky podpisov vytvorených knižnicou LTES a text tejto práce. Nosič je šrukturovaný nasledovne (spomenutý je len tu podstatný obsah):

- `LTES/Windows/app/` - aplikácia LTES (LTES.exe) spolu s knižnicou LTES (LTES.lib) a ďalšími potrebnými knižnicami.
- `LTES/Windows/src/` - zabalený projekt Visual Studio 2005 so zdrojovými kódmi knižnice a utility LTES
- `LTES/Linux/app/` - zdrojové kódy utility LTES pre OS Linux
- `LTES/Linux/src/` - zdrojové kódy knižnice LTES pre OS Linux
- `OpenSSL/src/` - zdrojové kódy knižnice OpenSSL 0.9.8c, na ktoré bol aplikovaný patch OpenTSA
- `OpenSSL/test/` - testovacie prostredie OpenSSL
- `thesis/` - text diplomovej práce vo formátoch PDF a DVI
- `example/` - príklady elektronických podpisov vytvorených pomocou utility LTES

Literatúra

[1] R. Housley: Cryptographic Message Syntax (CMS), RFC 3852, 2004

[2] European Telecommunication Standard Institute, 2006: Electronic signature formats for long term electronic signatures, ETSI TS 101 733 V1.6.3 (2005-09). ETSI.

[3] Pinkas, N.Pope: CMS Advanced Electronic Signatures (CAdES), RFC 5126, 2008.

[4] European Telecommunication Standard Institute, 2002: XML Advanced Electronic Signatures (XAdES), ETSI TS 101 903, ETSI.

[5] ITU-T X.680: Abstract Syntax Notation One (ASN.1): Specification of basic notation.

[6] ITU-T X.681: Abstract Syntax Notation One (ASN.1): Information object specification.

[7] ITU-T X.690: ASN.1 encoding rules: Specification of basic encoding Rules (BER), Canonical encoding rules (CER) and Distinguished encoding rules (DER).

[8] L. Dostálek, M. Vohnoutová (2006) : Velký průvodce infrastrukturou PKI a elektronickým podpisem, Computer Press, Brno, 2006.

[9] W3C: XML Signature Syntax, http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/ .

[10] D. Cooper a kol. : Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, 2008.

[11] C. Adams a kol. : Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), RFC 3161, 2001.