MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Digitální podepisování PDF souborů

DIPLOMOVÁ PRÁCE

Martin Henzl

Brno, jaro 2009

Prohlášení

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

Vedoucí práce: Ing. Mgr. Zdeněk Říha, Ph.D.

Poděkování

Chtěl bych poděkovat doktoru Zdeňku Říhovi a doktoru Petru Švendovi za pomoc a odborné konzultace.

Shrnutí

Tato práce se zabývá digitálním podepisováním PDF dokumentů pomocí čipových karet. Popisuje strukturu formátu PDF a způsoby provádění změn a vkládání nových objektů. Dále se zabývá digitálním podepisováním a integrací digitálního podpisu do struktury dokumentu PDF. Soustřeďuje se na použití hashovací funkce SHA-1 v kombinaci s šifrovacím algoritmem RSA s délkami klíčů 1024 a 2048 bitů. Práce dále seznamuje s čipovými kartami (angl. Smart Card) a jejich využitím pro digitální podepisování. Použitá technologie JavaCard umožňuje vytvořit vlastní program, zvaný aplet, který běží na kartě a provádí požadované operace. Aplet komunikuje s aplikací v počítači pomocí APDU datagramů. Tato práce se věnuje i programování apletů a APDU datagramům.

Klíčová slova

PDF dokument, Incremental update, Digitální podpis, RSA, Čipová karta, JavaCard, APDU datagram

Obsah

Seznam obrázků

Kapitola 1

Úvod

1.1 Úvod do problematiky

Jedním z nejrozšířenějších současných formátů pro přenos dokumentů je formát PDF (angl. Portable Document Format). Nabízí velké množství funkcí, které usnadňují orientaci v dokumentu, poskytují interaktivní prvky nebo nabízí zabezpečení dokumentu. Pro zajištění důvěrnosti dat je poskytována možnost šifrování dokumentu. Pro zaručení autentičnosti, integrity a nepopiratelnosti dokumentu poskytuje PDF možnost digitálního podepsání.

Digitální podepisování spadá do oblasti asymetrické kryptografie. K podepisování se používá dvojice klíčů - soukromý a veřejný. Soukromý klíč slouží k vytvoření podpisu a podepisující entita ho uchovává v tajnosti. Veřejný klíč se spojí s údaji o podepisujícím, čímž vznikne certifikát, který je zveřejněn a který slouží k tomu, aby kdokoli mohl ověřit platnost digitálního podpisu.

Soukromý klíč musí být bezpečně uchován v tajnosti. Toho lze docílit více způsoby, například zašifrováním nebo uložením na bezpečné místo. Problém bezpečného uchování klíče efektivně řeší čipové karty. Moderní kryptografické čipové karty poskytují kromě uchování páru klíčů také jejich generování přímo na kartě, dále pak nabízí různé kryptografické funkce, takže se dá podepisování provádět přímo na kartě a soukromý klíč nikdy nemusí opustit kartu, což velmi zvyšuje bezpečnost.

1.2 Motivace

Tato práce byla motivována požadavkem na vytvoření aplikace pro komerční využití. Má se jednat o program, který bude digitálně podepisovat naskenované dokumenty na *embedded* zařízení, ovládaném operačním systémem Linux. Toto zařízení bude fungovat jako skener, který dokumenty rovnou po naskenování podepíše. Podepisování bude probíhat na čipové kartě, takže soukromý klíč tuto kartu nebude muset vůbec opustit.

Kapitola 2

Formát PDF

PDF je jazyk určený k reprezentaci digitálních dokumentů, obvykle rozvržených do stránek, které mohou obsahovat text, obrázky, odkazy a další prvky. Cílem tohoto formátu je umožnit uživatelům jednoduchý a spolehlivý přenos digitálních dokumentů mezi zařízeními, nezávisle na prostředí, ve kterém byly vytvořeny nebo prolíženy. Je navržen tak, aby byl hardwarově nezávislý a přenositelný mezi různými operačními systémy. Tím je dosaženo toho, že libovolný dokument je možné zobrazit na všech zařízeních stejně.

Formát PDF nabízí široké spektrum funkcí, umožňuje textové vyhledávání, náhodný přístup k datům, podporuje záložky, odkazy, poznámky, interaktivní prvky a další. Dále umožňuje dokument komprimovat, zašifrovat nebo digitálně podepsat.

Vícestránkové dokumenty mohou být optimalizovány pro rychlé prohlížení na Internetu. Dokument může být načten tak, že se nejprve stáhnou data potřebná pro zobrazení požadované stránky a teprve potom se stahuje zbytek dokumentu. Další výhodou je možnost komprese dat, vhodná pro dokumenty poskytované na Internetu.

Jádrem PDF je pokročilý zobrazovací model odvozený z jazyka PostScript. Tento model umožňuje zobrazení dokumentu nezávisle na zařízení a na rozlišení. Oproti PostSriptu je ale PDF více strukturované a je optimalizované pro interaktivní prohlížení dokumentu. Navíc obsahuje objekty, které nejsou přímo součástí zobrazené stránky, ale jsou užitečné během interaktivního prohlížení. Jedná se o různé poznámky a hyperlinkové odkazy.

Ačkoli se z PDF vyvinul kvalitní interaktivní formát dokumentů, navíc velmi rozšířený a populární, je třeba si všimnout i jeho nedostatků. Formát PDF obsahuje chyby v návrhu a to především míchání syntaxe a sémantiky, což může vést k budoucím problémům.

Dalším nedostatkem je to, že dokumenty po převodu z jiného formátu nevypadají vždy úplně stejně jako předloha, především je to problém barev a fontů. Některé varianty formátu PDF, jako například PDF/X, však podle tvrzení autorů garantují přesnou kopii originálu.

Soubor ve formátu PDF není jednoduché zpracovat, protože obsahuje mnoho funkcí, které musí prohlížeč zvládat. Vytvoření kvalitního softwaru pracujícího s PDF vyžaduje hluboké znalosti o něm.

PDF nepracuje s odstavci, formátováním, záhlavím, zápatím, odsazením, dělením slov na konci řádků atd., což způsobuje značné komplikace při převodu dokumentu např. zpět do formátu Microsoft Word. Dále je tím znesnadněno porovnávání obsahu dvou dokumentů, neboť text není v souboru uložen sekvenčně nebo jako část věty nebo odstavce, ale ve fragmentech různě umístěných na stránce.

2.1 Historie a vývoj

PDF je formátovací jazyk, původně koncipovaný Johnem Warnockem, jedním ze zakladatelů společnosti Adobe Systems. První verze, 1.0, byla zveřejněna v roce 1993. Zpočátku nebyl tento formát tolik oblíbený, k pomalému rozšíření mezi uživatele přispělo především to, že jediný nástroj na tvorbu PDF dokumentů, Adobe Acrobat, byl komerční a lidé často volili jiné formáty, které byly použitelné zdarma. Dalším nedostatkem byla velikost souboru, která byla mnohem větší v porovnání s čistě textovým souborem, což nebylo vzhledem k tehdejším rychlostem internetového připojení zanedbatelné. Kromě toho samotné renderování takového dokumentu trvalo déle, než u jiných typů dokumentů, což bylo také znatelné díky tehdejšímu nižšímu výkonu počítačů.

Prohlížeč dokumentů PDF, Acrobat Reader, později Adobe Reader, byl již od začátku zdarma, což umožnilo komukoliv bezplatně tyto dokumenty prohlížet. Díky platformové nezávislosti, celkem spolehlivě stejnému zobrazení dokumentů na všech zařízeních, které tento formát podporovaly, a díky volně šiřitelnému prohlížeči se formát PDF stal postupně jakýmsi standardem v oblasti dokumentů určených k šíření v síti Internet.

Formát PDF byl navržen s ohledem na využití v síti Internet a právě díky velkému rozšíření Internetu v posledních zhruba deseti letech se stal tak populárním. Po první verzi z roku 1993 následovala řada dalších verzí, které přidaly do specifikace nové funkce. 29. ledna 2007 oznámila společnost Adobe Systems svůj záměr publikovat kompletní specifikaci PDF 1.7 jako mezinárodní standrd ISO. Od roku 1995 Adobe spolupracovalo s různými skupinami na vývoji technických specifikací určených k publikaci mezinárodní standadizační organizací ISO a dosáhlo standardizace několika specializovaných forem PDF, jako je např. PDF for Archive (PDF/A) nebo PDF for Exchange (PDF/X). Na jaře 2008 připravila společnost Adobe Systems Incorporated dokument ISO 3200, který vycházel z PDF 1.7 a který byl následně schválen technickou komisí. Formát PDF byl nakonec 1.července 2008 uvolněn jako otevřený standard a publikován jako ISO 32000-1:2008. Pro verzi 1.8 již Adobe žádnou specifikaci nevytváří, místo toho zveřejňuje dokumenty specifikující nové vlastnosti, které v normě ISO 32000-1:2008 nejsou.

verze PDF	verze Acrobatu	rok vydání
PDF 1.0	Acrobat 1.0	1993
PDF 1.1	Acrobat 2.0	1994
PDF 1.2	Acrobat 3.0	1996
PDF 1.3	Acrobat 4.0	1999
PDF 1.4	Acrobat 5.0	2001
PDF 1.5	Acrobat 6.0	2003
PDF 1.6	Acrobat 7.0	2005
PDF 1.7	Acrobat 8.0	2006
PDF 1.7, Adobe Extension Level 3	Acrobat 9.0	2008

Obrázek 2.1: Přehled verzí PDF

2.2 Struktura PDF

Na nejnižší úrovni se můžeme na soubor PDF dívat jako na sekvenci bajtů. Při dodržení syntaktických pravidel můžeme tyto bajty seskupit do celků, které jsou základní stavební jednotkou dokumentu PDF a nazývají se *objekty*.

Nezašifrovaný PDF soubor je možné vytvořit tak, že je reprezentován pouze viditelnými tisknutelnými znaky sady ANSI X3.4-1986 a „bílými znaky“. Obecně však dokument PDF může obsahovat libovolné bajty.

Strukturu dokumentu PDF můžeme sledovat ve dvou úrovních:

- **Struktura souboru** určuje, jak jsou *objekty* v PDF uloženy, jak se k nim přistupuje a jak jsou aktualizovány. Je nezávislá na sémantice *objektů*.
- **Struktura dokumentu** určuje, jak se základní *objekty* používají pro reprezentaci obsahu dokumentů, jako jsou jednotlivé stránky, fonty atd.

2.2.1 Objekty

Nejprve se podíváme na to, jaké typy objektů se v PDF používají a jaký mají tvar. Každý typ objektu lze jednoznačně rozeznat podle počátečního a koncového řetězce, který se skládá pouze ze znaků ASCII. Objekty mohou být i vnořené.

Specifikace PDF definuje tyto typy objektů: Boolean, Numeric, String, Name, Array, Dictionary, Stream, Null a Indirect.

Boolean Object
Reprezentuje logickou hodnotu *true* nebo *false*. V dokumentu PDF se vyskytuje jako klíčové slovo `true` nebo `false`.

Numeric Object
Číselný typ objektů reprezentuje buď celá nebo reálná čísla. Rozsah a přesnost reálných čísel jsou omezeny počítačem, na kterém se PDF právě zpracovává. Celá čísla se zapisují jako jedna nebo více číslic se znaménkem nebo bez něj, reálná čísla se zapisují jako jedna nebo několik číslic s desetinnou tečkou se znaménkem nebo bez něj.

String Object
Řetězce jsou objekty, které mohou uchovávat nula nebo více bajtů, jsou zapisovány dvěma způsoby:

- textový řetězec je uzavřen v kulatých závorkách
- řetězec je zapsán hexadecimálně a je uzavřen mezi znaky „<“ a „>“

Mezi kulatými závorkami mohou být uchovány libovolné tisknutelné znaky, kromě závorek „(“ a „)“ a zpětného lomítka. Pro uchování těchto a dalších speciálních symbolů jsou definovány escape sekvence. Například zpětné lomítko následované koncem řádku je chápáno jako pokračování řádku bez přerušení. Délka řetězce je omezena danou implementací.

Name Object
Objekty tohoto typu jsou symboly jednoznačně definované řetězcem znaků. Tento řetězec (jméno) může být libovolný, nemusí být nijak strukturovaný. V PDF tento objekt začíná symbolem „/“ následovaným jménem v textové podobě.
příklad: `/Root`

Array Object
Tyto objekty reprezentují jednorozměrná sekvenční pole, která mohou obsahovat jiné objekty. Na rozdíl od mnoha jiných jazyků mohou tato pole obsahovat současně objekty různých typů a to včetně jiných polí. Pole je v PDF uloženo jako sekvence objektů oddělených mezerou nebo jinými bílými znaky. Na začátku je pole uvozeno znakem „[“, na konci se nachází znak „]“.
příklad: `[100 3.14 false (text) /Atom]`

Dictionary Object
Objekty *dictionary*, neboli slovníky, jsou asociativní tabulky, které uchovávají dvojice objektů. První z dvojice je klíč, druhý je hodnota. Klíčem musí být na rozdíl od jazyka PostScript pouze objekt typu *name*, hodnotu může reprezentovat libovolný objekt a to včetně objektu *dictionary*. Na začátku slovníku jsou znaky „<<“, následují dvojice objektů oddělené bílými znaky, na konci je pak uzavírající značka „>>“.
příklad:
```
<< /Type /Example
   /SubType /DictionaryExample
   /Version 0.01
   /IntegerItem 12
   /StringItem (a string)
   /SubDictionary << /Item 0.4 >>
   >>
```

Slovníky jsou hlavními stavebními bloky dokumentů PDF. Udržují atributy komplexních objektů, jako jsou fonty, stránky apod. Klíčem bývá jméno atributu, druhá položka je pak hodnota.

Stream Object
Tento objekt umožňuje, podobně jako *string*, uchovávat řetězec bajtů. Není však omezena jeho délka a je tudíž vhodný pro ukládání dlouhých dat, jako např. obrázků. Data v tomto objektu mohou být složena z libovolných bajtů. Objekt *stream* má na začátku objekt *dictionary*, ve kterém jsou uloženy všechny potřebné informace o objektu, především délka dat. Za koncem objektu *dictionary* následuje klíčové slovo `stream`, konec řádku složený buď ze znaků CARRIAGE RETURN + LINE FEED nebo jen LINE FEED, za ním začínají data. Data jsou ukončena dalším koncem řádku a klíčovým slovem `endstream`. Povinné konce řádků se do délky dat nezapočítávají, neboť nejsou jejich součástí.
příklad: dictionary
```
stream
... nula nebo více bajtů ...
endstream
```

Null Object
V PDF se pro něj používá klíčové slovo `null`. V případě použití objektu typu *null* jako hodnoty v *dictionary* je to považováno za stejný případ, jako by tam tato položka vůbec nebyla. Podobně *indirect* odkaz na objekt, který neexistuje, je brán jako objekt typu *null*.

Indirect Object
Toto je typ, který se používá pro odkazování na jeden objekt z různých míst v dokumentu. Každý z předchozích typů objektů se může stát *indirect* objektem tak, že je mu přidělen unikátní identifikátor, uložený v hlavičce objektu. Pokud kdekoliv v dokumentu použijeme odkaz *indirect reference* s tímto identifikátorem, je to bráno tak, jako by se tento objekt v daném místě nacházel. Tento princip umožňuje použití jednoho objektu na více místech v dokumentu, aniž by byl na každém místě celý vypsán. Hlavně to ale umožňuje snadnou implementaci hierarchické *struktury dokumentu*.
Identifikátor se skládá ze dvou čísel:

- *object number* (přirozené číslo)
  Objekty obvykle bývají číslovány sekvenčně v rámci dokumentu, ale není to pravidlem.
- *generation number* (nezáporné celé číslo)
  Všechny *indirect* objekty v nově vytvořeném dokumentu by měly mít toto číslo rovno nule. Nenulová čísla se pak přidělují při provádění změn.

Indirect object je definován v PDF dokumentu pomocí *object number*, *generation number* a klíčového slova `obj`. Na konci objektu je pak klíčové slovo `endobj`.
příklad:
```
12 0 obj
  (text)
  endobj
```

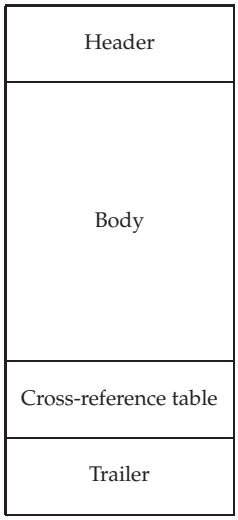
Na tento objekt se pak dá odkazovat pomocí tzv. *indirect reference* ve tvaru *object number*, *generation number* a znak „R“.
příklad: `12 0 R`

2.2.2 Struktura souboru

Soubor PDF se skládá ze čtyř základních prvků: *header*, *body*, *cross-reference table* a *trailer*. *Header*, neboli hlavička dokumentu, specifikuje verzi PDF, ve které byl dokument vytvořen. Tělo je složeno z objektů, které tvoří obsah dokumentu. Tabulka křížových odkazů *cross-reference* obsahuje informace o indirect objektech a konečně *trailer* určuje pozici *cross-reference* tabulky a některých významných objektů v dokumentu.

Header
První řádek souboru PDF je hlavička sestávající z pěti znaků „%PDF-“ následovaných číslem verze ve tvaru 1.N, kde N je číslice 0 až 7.

Body
Tělo souboru PDF se skládá ze sekvence *indirect* objektů, které reprezentují obsah dokumentu. Tyto *indirect* objekty se skládají z objektů základních typů, které reprezentují komponenty dokumentu, jako jsou jednotlivé stránky, fonty nebo obrázky. Od verze PDF 1.5 může také obsahovat objekty typu *Object Stream*.



Obrázek 2.2: Struktura souboru

Cross-reference
Tabulka křížových odkazů obsahuje informace, které dovolují přistupovat k *indirect* objektům v dokumentu bez nutnosti sekvenčního procházení. Tato tabulka by měla obsahovat jeden řádek pro každý *indirect* objekt, ve kterém je uvedeno číslo objektu a jeho offset v bajtech od začátku souboru. Každá sekce obsahující *cross-reference* tabulku začíná klíčovým slovem **xref**. PDF soubor může obsahovat více těchto sekcí, podrobněji to bude popsáno v části věnované *incremental update*.
Každá tabulka se skládá z jedné nebo více podsekcí. Každá podsekce obsahuje jeden nebo více řádků, které reprezentují jeden nebo několik po sobě jdoucích objektů. Na začátku každé podsekce je řádek specifikující *object number* prvního *indirect* objektu a počet po sobě jdoucích objektů.
příklad:
`28 5`
Tento řádek znamená, že v podsekci se vyskytují offsety objektů 28 až 32.

Následují jednotlivé záznamy ve formátu `nnnnnnnnnn ggggg n eol`, kde:
- *nnnnnnnnnn* je desetimístný offset v bajtech od začátku souboru
- *ggggg* je pětimístné *generation number*
- **n** klíčové slovo, pokud je to **n**, tento objekt se používá, pokud je to **f**, objekt je volný a nepoužívá se
- *eol* dvouznakový konec řádku

První položkou *cross-reference* tabulky by vždy měl být záznam s *generation number* 65535, nastavený jako volný (**f**).

V tomto příkladu obsahuje tabulka jedinou podsekci, která specifikuje objekty s čísly 0 až 5:

```
xref
0 6
0000000003 65535 f
0000000017 00000 n
0000000081 00000 n
0000000000 00007 f
0000000331 00000 n
0000000409 00000 n
```

V tomto příkladu obsahuje tabulka čtyři podsekce:

```
xref
0 1
0000000000 65535 f
3 1
0000025325 00000 n
23 2
0000025518 00002 n
0000025635 00000 n
30 1
0000025777 00000 n
```

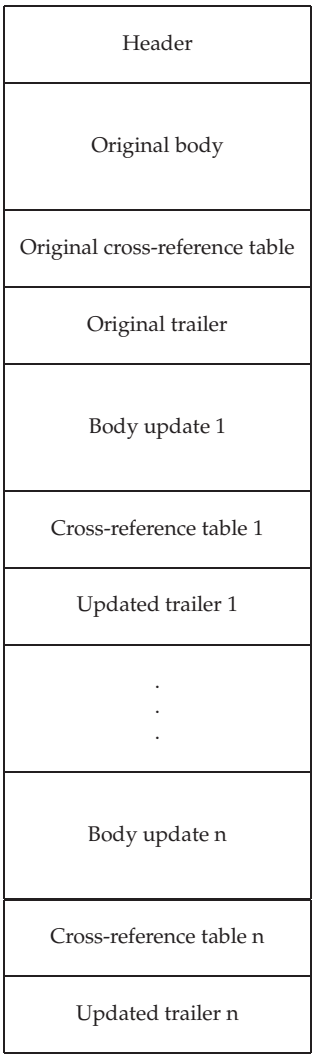
Trailer

Trailer umožňuje rychlé nelezení *cross-reference* tabulky a některých dalších důležitých částí dokumentu bez nutnosti procházet soubor sekvenčně. Prohlížeč PDF dokumentů by měl soubor začít číst od konce. Poslední řádek souboru obsahuje značku konce dokumentu - **%%EOF**. Dva předchozí řádky obsahují postupně klíčové slovo **startxref** a offset začátku tabulky *cross-reference*. Před slovem **startxref** se nachází *trailer dictionary*, na jehož začátku je klíčové slovo **trailer**. Položky *trailer dictionary* specifikují některé důležité vlastnosti dokumentu. Obsahuje tyto položky (dvojice klíč - hodnota):

- **Size** - celkový počet položek v tabulce *cross-reference*, včetně sekcí přidaných jako *incremental update*
- **Prev** - používá se pouze v případě *incremantal update*, určuje offset předchozí tabulky *cross-reference*
- **Root** - odkaz na kořenový objekt - *Document Catalog*
- **Encrypt** - používá se pouze pokud je dokument šifrován, odkaz na objekt definující šifrování
- **Info** - odkaz na objekt *Document Information Dictionary*
- **ID** - povinné, pokud je dokument šifrován, jedná se o identifikátor

2.2.3 Incremental update

Pro změny prováděné v PDF se může použít tzv. *incremental update*, který spočívá v tom, že se zachová celý původní obsah a na konec se přidají nové prvky *body*, *cross-reference* a *trailer*. Nové tělo dokumentu obsahuje v nové podobě ty objekty, které byly změněny. Při čtení takového dokumentu se berou v úvahu a zobrazují jen nejnovější verze všech objektů. Tímto způsobem je možné PDF dokument libovolně změnit. Dále je možné vrátit se k libovolnému stádiu vývoje dokumentu prostým odtržením nově přidaných prvků.



Obrázek 2.3: Incremental update

Tento způsob změny dokumentu mimo jiné dovoluje podepsat dokument a později k němu např. přidat nové stránky nebo ho změnit, přičemž pro původně podepsanou část zůstane podpis stále platný. Mohlo by se zdát, že pak nějaký útočník může takto podepsaný dokument změnit a uživatel to nepozná, jenomže uživatel uvidí, že dokument má platný podpis jen pro nějakou konkrétní verzi, a může si nechat zobrazit původně podepsanou verzi. Způsob změn *incremental update* to umožňuje udělat velmi snadno.

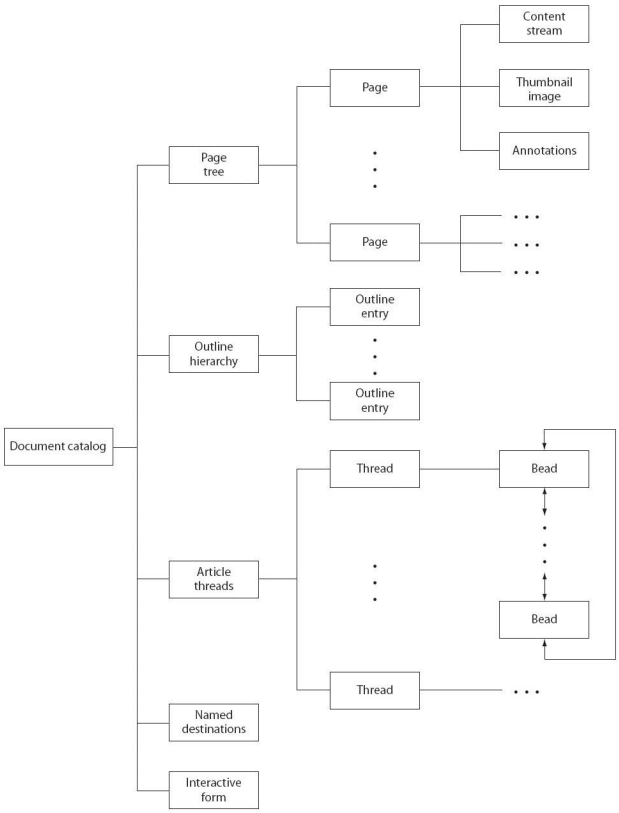
2.2.4 Struktura dokumentu

Na logické úrovni je dokument PDF tvořen hierarchií objektů. Všechny objekty v těle dokumentu jsou zapouzdřeny v *indirect* objektech, takže je možné na ně odkazovat. Pomocí referencí je vytvořena hierarchická struktura. Kořenem této hierarchie je *Document Catalog*. Stejně jako většina ostatních objektů v hierarchii je typu *dictionary*. Tyto objekty se skládají z jiných objeků, které už nebývají zapouzdřeny v *indirect* objektech, a které specifikují vlastnosti jednotlivých komponent dokumentu.

Document Catalog obsahuje odkazy na další objekty, které definují obsah dokumentu, dále informace o tom, jak má být dokument zobrazen na obrazovce, jako například jestli se při otevření automaticky zobrazí záložky a náhledy stránek a jestli se má jako první zobrazit jiná část než první stránka. *Document Catalog* například obsahuje položku `Pages`, která je kořenem stromu reprezentujícího stránky dokumentu. Odkazuje na jednotlivé stránky, které jsou reprezentovány objekty typu *dictionary* s názvem `Page`, které obsahují položky popisující atributy dané stránky. Stejným způsobem se dá najít cesta ke všem ostatním komponentám dokumentu.

Fyzicky mohou být objekty v souboru uloženy v libovolném pořadí, hierarchie je tvořena postupným odkazováním na objekty na nižší úrovni. Jelikož je *Document Catalog* kořenem hierarchie a je nutné u něj procházení či zobrazování dokumentu začít, je odkaz na něj uložen v části *trailer*, která již byla popsána a která se vždy fyzicky nachází na konci souboru.

Obrázek 2.4 znázorňuje strukturu dokumentu.



Obrázek 2.4: Struktura dokumentu PDF

Kapitola 3

Digitalní podpis

Digitální podpis je prostředek, který se používá v digitálních dokumentech pro zaručení autenticity, integrity a nepopiratelnosti. Je to číslo závislé na nějakém tajemství, které zná jen podepisující osoba, a také na samotném obsahu podepisovaného dokumentu. V případě nějaké pochybnosti o tom, zda někdo dokument podepsal či nikoli, at' už se jedná o popření skutečným podepsaným nebo o pokus o podvržení autorství cizí osobou, by mělo být možné ověřit nezávislou třetí stranou, zda byl dokument danou osobou podepsán, a to bez znalosti jejího tajemství (soukromého klíče).

Digitální podpis elektronického dokumentu by měl být ekvivalentí ručnímu podpisu papírového dokumentu. Na rozdíl od něj ale nabízí větší důvěryhodnost, neboť se nedá snadno zfalšovat a po podepsání už nelze do dokumentu nic přidat, aniž by byl podpis zneplatněn.

3.1 Zákon č. 227/2000 Sb., o elektronickém podpisu

Zákon č. 227/2000 Sb., o elektronickém podpisu specifikuje požadavky, které musí digitální podpis a s ním související subjekty splňovat, aby mohl být digiální podpis používán. V tomto zákoně se zavádí pojem elektronický podpis, zaručený elektronický podpis a elektronická značka.

Elektronický podpis jsou údaje v elektronické podobě, které jsou připojené k datové zprávě nebo jsou s ní logicky spojené a které slouží jako metoda k jednoznačnému ověření identity podepsané osoby ve vztahu k datové zprávě. [5]

Elektronickým podpisem tedy může být podle zákona i pouhé jméno autora v elektronické podobě.

Zaručený elektronický podpis je elektronický podpis, který splňuje následující požadavky [5]

- je jednoznačně spojen s podepisující osobou
- umožňuje identifikaci podepisující osoby ve vztahu k datové zprávě
- byl vytvořen a připojen k datové zprávě pomocí prostředků, které podepisující osoba může udržet pod svou výhradní kontrolou
- je k datové zprávě, ke které se vztahuje, připojen takovým způsobem, že je možno zjistit jakoukoliv následnou změnu dat

Elektronickou značkou se rozumí údaje v elektronické podobě, které jsou připojené k datové zprávě nebo jsou s ní logicky spojené a které splňují následující požadavky [5]

- jsou jednoznačně spojené s označující osobou a umožňují její identifikaci prostředictvím kvalifikovaného systémového certifikátu
- byly vytvořeny a připojeny k datové zprávě pomocí prostředků pro vytváření elektronických značek, které označující osoba může udržet pod svou výhradní kontrolou
- jsou k datové zprávě, ke které se vztahují, připojeny takovým způsobem, že je možné zjistit jakoukoli následnou změnu dat

Rozdíl mezi elektronickým podpisem a elektronickou značkou je ten, že v případě použití elektronického podpisu se předpokládá, že autor podepsal dokument s plným vědomím a souhlasí s jejím obsahem, zatímco elektronická značka bývá použita při automatizovaném podepisování, nekladou se tedy nároky na nepopiratelnost.

Digitální podpis založený na asymetrické kryprografii splňuje podmínky zaručeného elektronického podpisu definovaného zákonem.

3.2 Mechanizmus digitálního podpisu

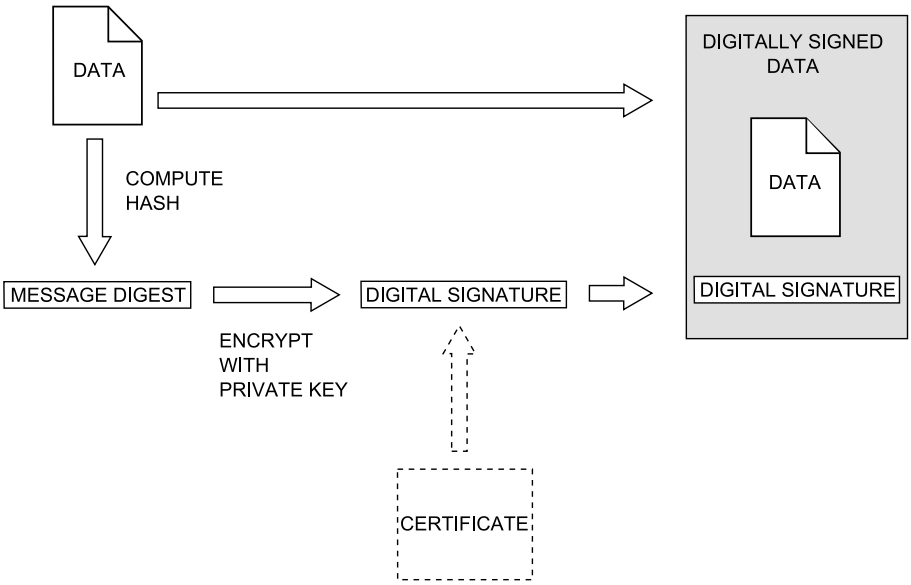
Digitální podepisování je založeno na asymetrické kryptografii. Používají se dva klíče, jeden pro podpis, druhý pro ověření - verifikaci. Pokud chce osoba A podepsat zprávu M, musí mít nejprve k dispozici pár navzájem korespondujících klíčů - soukromý klíč K_s a veřejný klíč K_v. Pokud chce libovolná osoba B ověřit, že zprávu M podepsala osoba A, potřebuje k tomu veřejný klíč K_v osoby A.

Pro vytvoření a ověření digitálního podpisu jsou zapotřebí následující algoritmy:
$H(M)$ ... kryptografická hash funkce, počítá hash zprávy M
$S_{Ks}(X)$ ... algoritmus, který podepíše zprávu X klíčem K
$V_{Kv}(X)$ ... verifikační algoritmus, který ověří, zda byla zpráva X podepsána klíčem K, přičemž platí, že $V_{Kv}(S_{Ks}(X)) = X$

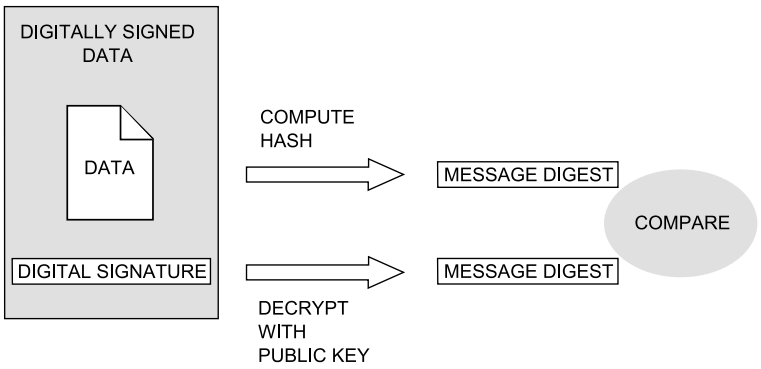
Vytvoření digitálního podpisu
Entita A vytvoří digitální podpis DS tak, že nejprve vypočítá hash celé zprávy a tu potom „zašifruje".
$DS = S_{Ks}(H(M))$

Ověření digitálního podpisu
Libovolná entita B ověří digitální podpis zprávy M_1 pomocí veřejného klíče K_v entity A tak, že tímto klíčem dešifruje podpis, z toho získá hash podepsaného dokumentu, kterou porovná s hashí zprávy M. Pokud jsou tyto hashe shodné, je podpis platný, v opačném případě je neplatný.
$V_Kv(S_{Ks}(H(M))) == H(M_1)$?

Pokud podepisující osoba klíče nemá, musí je vygenerovat. Algoritmus na vytváření klíčů musí vytvářet klíče, které jsou co nejvíce podobné náhodnému shluku bitů, často se k tomu používá generátor náhodných čísel. Náhodnost bývá získávána z různých zdrojů, například z přesných časů uživatelových úhozů na klávesnici nebo je možné použít speciální hardware. Po vytvoření si uživatel soukromý klíč uchová v tajnosti a veřejný klíč zveřejní tak, aby kdokoliv, kdo bude potřebovat ověřit podpis, měl tento klíč k dispozici a měl jistotu, že je



Obrázek 3.1: Schéma podepisování



Obrázek 3.2: Schéma verifikace

to skutečně veřejný klíč tohoto uživatele a že není podvržený. Pokud by totiž uživatel B měl k dispozici podvržený klíč K_v2, nemohl by ověřit podpis, naopak podvržený podpis by mohl být ověřen jako platný. Pokud tedy uživatel A nemůže předat veřejný klíč bezpečnou cestou, musí se důvěryhodnost klíče zaručit jinou cestou. O zajištění důvěryhodnosti veřejných klíčů se starají důvěryhodné třetí strany - certifikační autority (CA).

3.3 Public Key Infrastructure

PKI (Public Key Infrastructure) je soubor hardwaru, softwaru, lidí, zásad a procedur nutných k vytvoření, správě, uchování, distribuci a revokaci certifikátů veřejných klíčů. [9]

Tato infrastuktura v sobě zahrnuje certifikační autority (CA). Certifikační autorita je organizace, která funguje jako důvěryhodná třetí strana u asymetriké kryptografie. Poskytuje provázání mezi veřejným klíčem osoby A a identitou osoby A. Toto provázání se nazývá certifikát. Certifikát je veřejný klíč osoby A podepsaný soukromým klíčem certifikační autority. CA tyto certifikáty vydává po ověření identity osoby A, čímž zaručuje, že klíč patří té pravé osobě. Aby bylo možné ověřit certifkát, je potřeba veřejný klíč certifikační autority a také důkaz, že tento klíč CA není podvržený. K tomu slouží kořenový certifikát CA, což je certifikát, který si CA sama podepsala nebo který podepsala jiná CA. Uživatelé mají ve svých počítačích seznamy kořenových certifikátů certifikačních autorit, kterým důvěřují. Některé kořenové certifikáty jsou již uloženy v novém operačním systému, další si může uživatel přidávat.

Alternativou k PKI je tzv. Web of Trust, neboli Síť důvěry. V tomto systému nejsou žádné certifikační autority, certifikáty si uživatelé ověřují a podepisují navzájem. Tento systém používá například PGP (Pretty Good Privacy).

3.4 Algoritmus RSA

Pro digitální podepisování se používají různé algoritmy, ve své práci používám podepisovací algoritmus RSA, proto se mu budu nyní věnovat podrobněji.

Algoritmus RSA je algoritmus asymetrické kryptografie, který je navržen tak, že může být použit pro šifrování i digitální podepisování. Vznikl v roce 1977. Název RSA získal algoritmus podle iniciál svých autorů, kterými jsou Ron Rivest, Adi Shamir a Len Adleman.

Šifra RSA je založena na matematickém problému faktorizace velkých čísel.
Prostor zpráv i prostor šifer jsou v RSA $Z_N = \{0, 1, 2, ..., n-1\}$, kde $n = pq$ je součin dvou náhodných prvočísel p a q. Protože je šifrovací transformace bijekce, může být digitální podpis vytvořen prohozením šifrování s dešifrováním. Digitální podpis vznikne dešifrováním hashe zprávy, ověřit se dá opětným zašifrováním. RSA je determinisický algoritmus a pokud podepisujeme přímo data a nikoliv hash, obdržíme tato data při ověřování.

3.4.1 Generování klíče pro RSA algoritmus

Každá entita si vytváří soukromý a s ním korespondující veřejný klíč tímto postupem:

1. Nejpve zvolí dvě velká náhodná prvočísla p a q, obě přibližně stejně velká.
2. Vypočítá $n = pq$ a hodnotu Eulerovy funkce $\phi = (p-1)(q-1)$.
3. Zvolí náhodné celé číslo e takové, že $1 < e < \phi$ a největší společný násobek $gcd(e, \phi) = 1$.

4. Použije rozšířený Euklidův algoritmus a vypočítá inverzi d čísla e, která je jednoznačná, $1 < d < \phi$ tak, že $ed \equiv 1(mod\phi)$.

5. Veřejným klíčem entity a je (n, e), soukromým klíčem je d.

3.4.2 Generování digitálního podpisu

Entita A podepisuje zprávu $m \in M$ pomocí svého páru klíčů takto:

1. Spočítá celé číslo $\tilde{m} = R(m)$ z oboru hodnot $[0, n-1]$.

2. Spočítá $s = \tilde{m}^d$ mod n.

3. Digitálním podpisem je s.

3.4.3 Ověření digitálního podpisu

Jakákoliv entita B může ověřit podpis entity A a získat zprávu m z podpisu tímto způsobem:

1. Získá autentický veřejný klíč (n, e) entity A.

2. Spočítá $\tilde{m} = s^e$ mod n.

3. Ověří, že $\tilde{m} \in M_R$. Pokud ano, podpis je platný, pokud ne, podpis je zamítnut.

4. Obnoví zprávu $m = R^{-1}(\tilde{m})$.

3.4.4 Důkaz korektnosti algoritmu

Jestliže s je digitální podpis zprávy m, pak $s \equiv \tilde{m}^d$ mod n, kde $\tilde{m} = R(m)$. Jelikož $ed \equiv 1$ (mod ϕ), $s^e \equiv \tilde{m}^{ed} \equiv \tilde{m}$ (mod n). Konečně $R^{-1}(\tilde{m}) = R^{-1}(R(m)) = m$.

Kapitola 4

Digitální podpisy v PDF

Dokumenty PDF umožňují vložení digitálního podpisu. Tento podpis je buď možné zobrazit přímo na stránce dokumentu nebo k němu uživatel přistupuje přes menu prohlížeče. Zobrazování digitálního podpisu na stránce dokumentu je záležitostí pouze grafickou a nikoli kryptografickou, vyžaduje hlubší znalosti práce s grafickými objekty v PDF a na funkčnost nemá vliv, nebudu se tedy tímto problémem dále zabývat. Cílem bude digitální podepsání dokumentu tak, aby ho bylo možné ověřit jakýmkoli prohlížečem, který to umožňuje. Takovým prohlížečem je například Adobe Reader, který se k prohlížení PDF používá nejčastěji a který digitální podpisy standardně podporuje.

Digitální podpis v PDF se skládá z několika elementárních objektů, které tvoří popis podpisu i samotná data podpisu. Uchovává informace o podepisující entitě a o stavu dokumentu v době podepsání.

Podpis v dokumentu může být buď čistě matematický nebo to může být biometrická forma identifikace, jako např. rukou psaný podpis, otisk prstu nebo obraz sítnice. Každý použitý specifický typ autentizace by měl být implementován speciálním softwarovým modulem zvaným *signature handler*.

ISO 3200 podporuje dvě operace s digitálními podpisy: vkládání digitálního podpisu a pozdější ověřování platnosti. Podepisující software si musí zjistit, zda není certifikát, který se má použít, revokován. Podobně se musí zjistit informace o revokaci pro ceký certifkační řetěz a to ještě před podepsáním.

4.1 Způsoby provedení změn v dokumentu PDF

Digitální podpis je možné do dokumentu PDF vložit dvěma způsoby.

- **Standard update**
  Pokud do dokumentu takto přidáváme data, v našem případě digitální podpis, měníme obsah celého dokumentu. Existující objekty se nahrazují novými, přidávají se nově vkládané objekty. Potom je nutné přidat odkazy na nové objekty do tabulky *cross-reference* a aktualizovat všechny offsety v dokumentu.

- **Incremental update**
  Na rozdíl od předchozí metody se nemusí celý dokument přepisovat, k původnímu dokumentu se pouze přidají nové objekty, nové verze změněných objektů z původního dokumentu, vytvoří se nová tabulka *cross-reference* a *trailer*. Tento způsob byl popsán v kapitole „Formát PDF“.

Standardně se pro digitální podpisy používá incremental update, neboť umožňuje snadné vrácení dokumentu do stavu před podepsáním.

4.2 Integrace digitálního podpisu do struktury dokumentu PDF

Informace o podpisu i vlastní data podpisu by měly být uloženy v *signature dictionary*, což je objekt typu *dictionary*. Tímto objektem definujeme základní vlastnosti podpisu. Položky, které se v něm nastavují, jsou uvedeny v tabulce 4.1. Položky, které jsou povinné, jsou označeny, ostatní jsou nepovinné, nemusí se tedy nastavovat.

Objekt *signature dictionary* je potřeba začlenit do struktury dokumentu. Struktura dokumentu je hierarchická, podíváme se postupně na všechny objekty, které je potřeba pro správnou funkci podpisu vložit. Budeme postupovat od kořenového objektu *document catalog* směrem k objektu *signature dictionary*.

Jako první je potřeba mít v dokumentu interaktivní formulář. Pokud tam zatím žádný není, musí se vytvořit nový. Jedná se o objekt *AcroForm*. Na tento objekt musíme mít odkaz ve slovníku *document catalog*. Interaktivní formulář *AcroForm* je množina polí, ktará slouží k interaktivnímu shromažďování informací od uživatele. Těchto polí může být v dokumentu libovolné množství. Každé pole je definováno objektem *field dictionary*.

Dále se musí vytvořit *signature field*, což je *field dictionary* s přidanými některými atributy. Atributu `FT` musí být nastavena hodnota `Sig`. Na tento objekt bude odkazovat atribut `Fields` objektu *AcroForm*.

Nakonec se musí nastavit atribut „V“ pole *signature field* tak, aby odkazoval na pole *signature dictionary*, čímž se dokončí cesta od kořenového objektu.

Schéma integrace podpisu do struktury dokumentu je vidět na obrázku 4.2.

4.3 Podporované algoritmy

Specifikace PDF dovoluje vkládat digitální podpisy podle standardu PKCS#1 nebo PKCS#7.

- **PKCS#1** - Tento standard podporuje RSA jako podepisovací algoritmus, dále pak dovoluje použití několika různých hashovacích funkcí, jakými jsou SHA-1 a MD5. PDF však pro podepisování podle PKCS#1 dovoluje použít pouze algoritmus RSA s hash funkcí SHA-1. Hodnota **SubFilter** musí být v *signature dictionary* nastavena na **adbe.x509.rsa_sha1**. Řetězec certifikátů je pak uložen v položce **Cert**.
- **PKCS#7** - V případě použití tohoto standardu bude položka **Contents** obsahovat binární data PKCS#7, obsahující podpis, a bude kódována jako struktura DER. **SubFilter** může být **adbe.pkcs7.detached** nebo **adbe.pkcs7.sha1**.
  Objekt PKCS#7 by měl obsahovat časové razítko jako nepodepsaný atribut a revokační informace jako podepsaný atribut. Jelikož revokační informace jsou podepsaný atribut, musí podepisující software ještě před podepsáním sestavit celý cetrifikační řetězec a získat všechny informace o revokaci.

Tabulka 4.3 ukazuje, které algoritmy je možné použít při konkrétních nastaveních položky **SubFilter**. Použitý algoritmus je pak specifikován přímo ve struktuře PKCS#1 nebo PKCS#7, v *signature dictionary* se neuvádí.

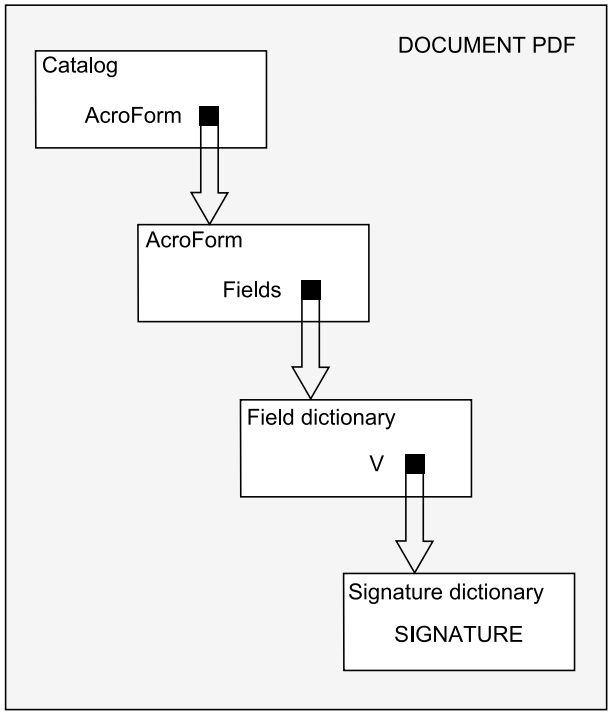
4.4 Vkládání a ověřování digitálních podpisů v PDF

Digitální podpis se vytvoří z celého dokumentu nebo jen z části, rozsah podepisovaných oblastí je dán položkou **ByteRange** která je v dokumentu uložena (v tzv. *signature dictionary*).

Klíč	Typ	Hodnota
Type	name	Typ PDF objektu, který tento *dictionary* objekt popisuje; v případě digitálního podpisu je to **Sig**.
Filter	name	*(povinný)* Jméno preferovaného handleru, který se má použít pro ověřování podpisu. Pokud není nastavena také položka **Prop_Build**, je to také jméno handleru, který byl použit pro vytvoření podpisu. Prohlížeč může použít pro ověření jiný handler, ten ale musí podporovat filtr specifikovaný položkou **SubFilter**. Podepisovací handler může být např. **Adobe.PPKLite**, **Entrust.PPKEF**, **CICI.SignIt** nebo **VeriSign.PPKVS**
SubFilter	name	Název, který udává způsob kódování dat podpisu a informací o klíči v *signature dictionary*. Prohlížeč může k ověření podpisu použít libovolný handler, který tento formát podporuje. Pro digitální podpisy asymetrické kryptografie by měla být použita jedna z následujících hodnot: **adbe.x509.rsa_sha1**, **adbe.pkcs7.detached** nebo **adbe.pkcs7.sha1**. Výrobci si mohou nadefinovat nové filtry, jejich názvy by měly začínat jménem výrobce.
Contents	byte string	*(povinný)* Hodnota podpisu. Pokud je použita položka **ByteRange**, hodnota bude hexadecimální řetězec, ohraničený znaky „<“ a „>“, reprezentující hodnotu digitálního podpisu, uložený v objektu *string*. **Contents** obsahuje data ve formátu PKCS#1 nebo PKCS#7 kódované DER. Protože délka objektu PKCS#7 není dopředu známa, pro tuto položku se alokuje místo ještě předtím, než se vypočítá podpis. Nevyužité místo se potom vyplní nulami až do konce stringu ke koncové značce „>“.
Cert	array or byte string	*(povinný pokud **SubFilter** je **adbe.x509.rsa_sha1**)* Reprezentuje certifikační řetěz certifikátů X.509 použitý pro podepisování a verifikaci. Je to buď objekt *array*, pokud je certifikátů víc a tvoří pole objektů *string*, nebo pouze objekt *string*, který reprezentuje jediný certifikát. První certifikát v poli je ten, který byl použit pro podepsání, ostatní slouží k ověření autenticity tohoto certifikátu.
ByteRange	array	*(povinný)* Pole dvojic celých čísel popisující všechny intervaly, které tvoří data, která se podepisují. První číslo z dvojice udává offset prvního bajtu intervalu, druhé číslo udává délku intervalu. Intervaly obvykle pokrývají celý dokument kromě vlastního digitálního podpisu. Data pokrytá těmito intervaly jsou podepsána a je tedy i v případě nedůvěryhodného certifikátu zaručena jejich integrita.

Reference	array	Pole objektů *signature reference dictionary*, které specifikují určité transformace.
Changes	array	Pole tří čísel, které specifikují změny, které byly s dokumentem provedeny mezi předchozím a tímto podepsáním. První číslo udává počet změněných stránek, druhé udává počet změněných polí a třetí udává počet polí, která byla vyplněna.
Name	text string	Jméno osoby nebo jiného subjektu, který dokument podepsal. Používá se pouze tehdy, když tyto informace nelze získat z podpisu (z certifikátu podepisující osoby).
M	date	Čas, kdy byl dokument podepsán. Používá se pouze tehdy, když tuto informaci nejde získat z podpisu (z certifikátu podepisující osoby).
Location	text string	Místo, kde byl dokument podepsán.
Reason	text string	Důvod podepsání.
ContactInfo	text string	Kontaktní informace na podepisujícího.
R	integer	Verze handleru, který byl použit k podepsání.
V	integer	Verze formátu *signature dictionary*. Pokud je hodnota rovna 1, bere se při validaci podpisu v potaz **Reference** dictionary.
Prop_Build	dictionary	Může být použito pro uložení údajů o prostředcích počítače, který dokument podepsal. Jedná se například o jméno handleru, který byl použit pro podepsání, operační systém apod.
Prop_AuthTime	integer	Počet sekund od doby, kdy byl podepisující naposledy autentizován.
Prop_AuthType	name	Metoda, která by měla být použita k autentizaci podepisujícího, může to být PIN, heslo nebo otisk prstu.

Obrázek 4.1: Signature dictionary



Obrázek 4.2: Schéma integrace podpisu do struktury dokumentu PDF

	adbe.pkcs7.detached	adbe.pkcs7.sha1	adbe.x509.rsa_sha1
Message Digest	SHA1 (PDF 1.3) SHA256 (PDF 1.6) SHA384 (PDF 1.7) SHA512 (PDF 1.7) RIPEMD160 (PDF 1.7)	SHA1 (PDF 1.3)	SHA1 (PDF 1.3) SHA256 (PDF 1.6) SHA384 (PDF 1.7) SHA512 (PDF 1.7) RIPEMD160 (PDF 1.7)
RSA	až 1024 bitů (PDF 1.3) až 2048 bitů (PDF 1.5) až 4096 bitů (PDF 1.5)	až 1024 bitů (PDF 1.3) až 2048 bitů (PDF 1.5) až 4096 bitů (PDF 1.5)	až 1024 bitů (PDF 1.3) až 2048 bitů (PDF 1.5) až 4096 bitů (PDF 1.5)
DSA	až 4096 bitů (PDF 1.6)	až 4096 bitů (PDF 1.6)	není podporováno

Obrázek 4.3: Podporované algoritmy pro různé hodnoty **SubFilter**

Pokud se podepisuje celý dokument, přidají se nejprve všechny objekty jako incremental update, přičemž místo pro samotná data podpisu zůstane prázdné. Toto místo má velikost odhadované délky podpisu. Potom se nastaví **ByteRange** tak, aby zahrnovalo celý dokument kromě dat podpisu. Z oblasti definované pomocí **ByteRange** se vypočítá digitální podpis a ten se uloží do prázdného místa.

Ověření podpisu probíhá tak, že se spočítá hash dat v oblasti definované prvkem **ByteRange** a porovná se s hashí získanou z digitálního podpisu pomocí veřejného klíče podepisující entity. Pokud se rovnají, je podpis platný a je ověřeno, že dokument nebyl od aplikace podpisu změněn. V opačném případě je buď podpis podvržený nebo byl dokument změněn, v každém případě je podpis neplatný.

Kapitola 5

Smart Cards

Čipové karty, anglicky „Smart Cards“, jsou plastové karty se zabudovaným integrovaným obvodem, který je schopen provádět operace, ke kterým je karta určena. Čipové karty mohou obsahovat buď jen paměť pro uložení dat nebo jsou vybaveny i mikroprocesorem. Čipové karty se také dělí na kontaktní a bezkontaktní.

Kryptografické čipové karty jsou vybaveny mikroprocesorem a bývají kontaktní. Nabízejí různé algoritmy pro generování náhodných čísel, hashování, šifrování a podepisování. Je například možné použít kartu podporující standard PKCS#11, který umožňuje aplikacím běžícím na počítači provádět některé operace na kryptografickém hardwaru, v tomto případě na čipové kartě. Další možností je použít modernější technologii, kartu JavaCard. Pro tuto kartu je možné psát vlastní aplikace, které se do ní nahrají. S čipovými kartami se komunikuje pomocí datagramů APDU.

5.1 Normy

Norma **ISO7816-1** definuje fyzické vlastnosti integrovaného obvodu, jako jsou limity pro Rentgenové a UV záření, elektromagnetické pole, teplotu atd. Dále definuje vlastnosti karet například při ohýbání, což má zaručit bezproblémou funkčnost karty po dobu její životnosti. Tato norma je důležitá pro výrobce karet.

Norma **ISO7816-2** definuje velikost a umístění kontaktů. Zahrnuje standardy o počtu, funkci a pozici elektrických kontaktů.
Čipové karty mají 8 kontaktů, označených C1 až C8. Ne všechny jsou ale propojeny s mikroprocesorem, některé zůstávají nevyužity.
Čipové karty mají tyto elektrické kontakty (postupně od C1 do C8): Vcc, RST, CLK, RFU, GND, Vpp, I/O, RFU. *(RFU = reserved for future use)*

Norma **ISO7816-3** popisuje elektrické signály a přenosové protokoly čipových karet.

- Protokol T=0 - byte-oriented, navržen tak, aby byl co nejjednodušší a aby měl malé nároky na paměť. Detekce chyb jen na úrovni paritních bitů. Používá se v GSM.
- Protokol T=1 - asynchronní, half-duplex, blokový.

Norma **ISO7816-4** specifikuje příkazy a odpovědi, přenášené na kartu a zpět, dále metody přístupu k souborům a datům na kartě a algoritmům, které karta poskytuje.

5.2 APDU

Komunikace s čipovou kartou probíhá pomocí komunikačních datagramů APDU (Application Protocol Data Unit). Struktura APDU je definována normou ISO 7816. Hlavička APDU datagramu posílaného kartě umožňuje specifikovat aplikaci na čipové kartě, pro kterou je

datagram určen. Kromě hlavičky může APDU obsahovat až 255 bajtů dat. Datagramy ve směru z karty do počítače mohou nést až 256 bajtů dat a na konci mají vždy status, který má délku 2 bajty.

Hlavička APDU obsahuje vždy tyto bajty (v tomto pořadí): CLA - instruction class, INS - instruction number, P1, P2 - volitelná data
Pokud datagram obsahuje nějaká data, hlavička obsahuje ještě jeden bajt - Lc - udávající délku dat, která následují.
Na konci APDU datagramu může být ještě jeden bajt - Le - určující očekávanou délku výstupních dat, která přijdou jako odpověď.

APDU datagram posílaný ve směru z karty do PC se skládá z Le bajtů dat a dvou bajtů, které specifikují stav provedené operace. Pokud byla operace provedena a vše je v pořádku, bude stav `0x90 0x00`. Pokud dojde k nějaké chybě, bude ve stavových bajtech odeslán kód této chyby.

5.3 JavaCard

Jednou z moderních technologií čipových karet je JavaCard. Tato technologie nabízí bezpečné prostředí pro běh aplikací na čipových kartách, je platformově nezávislá a je kompatibilní se všemi standardy čipových karet. Aplikace pro tyto karty jsou napsány v jazyce Java a nazývají se aplety. Zdrojové kódy apletů je možné překládat standardními překladači jazyka Java, výsledné .class soubory se zkonvertují do formátu určeného pro kartu a nainstalují se na čipovou kartu. Použití standardního jazyka Java usnadňuje vývoj a testování aplikací. Dalšími výhodami jsou bezpečnost a možnost implementace několika aplikací na jedné kartě.

Kapitola 6

Program na podepisování PDF pomocí čipové karty

6.1 Cíle

Cílem bylo navrhnout a implementovat program, který bude digitálně podepisovat PDF soubory pomocí čipové karty. Vzhledem k zamýšlenému využití této aplikace na *embedded* zařízeních bylo potřeba omezit použití externích knihoven a přizpůsobit se požadavkům na použitý operační systém a programovací jazyk. Aplikace musí běžet pod operačním systémem Linux a měla by být napsána nejlépe v jazyce C++.

Použití aplikace by mělo být takové, že program poběží na *embedded* zařízení, které bude skenovat dokumenty, z těchto dokumentů vytvářet PDF soubory a pomocí tohoto programu a soukromého klíče uloženého na čipové kartě tyto dokumenty podepisovat. Tím bude kromě zajištění integrity dokumentu možné zpětně zjistit, kdo daný dokument vytvořil a podepsal. Na způsob zobrazení podpisu v dokumentu nejsou kladeny žádné požadavky, je tedy možné vytvořit pouze neviditelný podpis, který se na stránce dokumentu nezobrazuje, ale ke kterému se dá dostat přes menu prohlížeče.

Aplikace dostane na vstup PDF soubor, jejím úkolem bude tento soubor syntakticky rozebrat a provést v dokumentu úpravy podle normy ISO 32000-1:2008 tak, aby bylo možné vložit digitální podpis. Dále musí program pomocí čipové karty vytvořit digitální podpis, který vloží do dokumentu. Podepsaný dokument pak uloží na disk. Soukromý klíč z důvodu bezpečnosti nesmí opustit čipovou kartu, podepisování tedy musí probíhat přímo na kartě.

Čipová karta musí být naprogramována tak, aby dokázala příchozí data podepsat algoritmem RSA pomocí soukromého klíče, uloženého na kartě. Kromě toho také musí umět vytvořit pár klíčů (soukromý a veřejný) a soukromý umět exportovat z karty, aby se z něj dal vytvořit certifikát, který bude sloužit k ověřování podpisů.

6.2 Návrh řešení

Aplikace bude spouštěna pro každý podpis zvlášť. Parametry předávané programu budou zahrnovat PIN, jméno souboru, v němž je podepisovaný dokument, jméno výstupního podepsaného souboru, jméno konfiguračního souboru, v němž se nastavují atributy, které se nemění moc často a nejsou tajné. V tomto konfiguračním souboru se kromě jména uživatele, důvodu podepsání, podepisovacího algoritmu apod. nastavuje jméno souboru, ve kterém je uložen certifikát podepisujícího uživatele ve formátu DER.

Na podepisujícím zařízení se musí pravidelně aktualizovat CRL (Certificate Revocation List), aby mohly být revokované certifikáty zablokovány a nemohly se použít pro podepisování. Tato aplikace se o revokaci nestará, načítá certifikát z definovaného souboru. Pokud je certifikát revokován, nesmí být této aplikaci poskytnut.

Pro vytvoření funkční aplikace bylo potřeba zvolit konkrétní vlastnosti digitálního podpisu. Jedná se o hashovací a podepisovací algoritmus, dále pak o normu, podle které se

bude podpis vytvářet. Specifikace PDF nabízí u každého z těchto atributů několik variant, které však dohromady dávají velké množství kombinací, některé nesmyslné nebo velmi nesnadno implementovatelné. Cílem této práce není vytvořit univerzální podepisovací nástroj, nýbrž ukázat způsob, jakým lze podepisování provádět s pomocí čipové karty s ohledem na spolehlivost a bezpečnost a vytvořit nástroj, který dokáže podepisovat.

Podepisování se tedy bude provádět pomocí hashovací funkce SHA-1 a asymetrického kryptografického algoritmu RSA. Celý podpis se pak bude vytvářet podle normy PKCS#1.

Hash dokumentu se může počítat programem v počítači, nemusí se počítat na kartě. Bez ztráty bezpečnosti tak dosáhneme výrazného zrychlení, protože se na čipovou kartu nebude muset posílat celý dokument, ale jen jeho mnohem menší hash.

Podepisování pomocí RSA provádí čipová karta. Pro tento účel byla zvolena karta JavaCard od firmy Gemalto, která nabízí RSA s délkou klíče až 2048 bitů. Pro podepisování pomocí RSA bude vytvořen Java aplet, který poběží na čipové kartě a který bude ke komunikaci s programem používat APDU (Application Protocol Data Unit).

Práce programu by se dala rozdělit do několika fází, přičemž v každé z nich bude prováděna jen určitá činnost. Seznam těchto fází pak může vypadat takto:

- **Syntaktická analýza dokumentu**
- **Provedení úprav - příprava na vložení digitálního podpisu**
- **Vypočítání hashe požadované části připraveného dokumentu**
- **Podepsání hashe čipovou kartou**
- **Vložení digitálního podpisu do připraveného dokumentu**

Při analýze dokumentu se musí provádět syntaktická analýza. Hledání počátečních značek požadovaných objektů by bylo sice rychlejší a jednodušší, to však není možné kvůli možnosti vnořených objektů a především kvůli tomu, že značky jsou textové a stejný řetězec by tedy mohl být náhodou obsažen i v textu dokumentu. Syntaktická analýza eliminuje nebezpečí chybného zpracování.

Program provádí *incremental update*. Je tedy nutné najít v dokumentu objekty, které se musí kvůli digitálnímu podpisu změnit, kopie těchto objektů upravit a vložit na konec dokumentu, přidat nové objekty a na závěr vytvořit tabulku *cross-reference*. Jelikož vkládáme neviditelný podpis, tak jediným objektem, který je nutné modifikovat, je objekt *Catalog*. Do tohoto objektu je třeba vložit odkaz na AcroForm. Vytvořit se musí objekty *AcroForm*, *Field dictionary* a *Signature dictionary*. Nová tabulka *cross-reference* musí obsahovat offsety všech těchto objektů. Do objektu *Signature dictionary* se musí vložit certifikát nebo řetězec certifikátů. Významnou částí této fáze je také správné nastavení **ByteRange** v *Signature dictionary*. Obsahuje dva intervaly, od začátku souboru po začátek podpisu a od konce podpisu po konec souboru. Kdyby byl použit podpis podle normy PKCS#7, tak by délka podpisu nebyla dopředu známa, musela by se tedy odhadnout a přebývající volné místo by se pak vyplnilo znaky nula, jejichž hexadecimální hodnota je `0x30`. U PKCS#1 tento problém není, protože podpis neobsahuje certifikáty.

Z oblasti definované prvkem **ByteRange** se vypočítá hodnota hash, která se pošle na čipovou kartu.

Algoritmus RSA bude aplikovat čipová karta. Karta bude mít tyto funkce:

- Generování páru klíčů (soukromého a veřejného) a přenos veřejného klíče z karty
- Podepisování algoritmem RSA za pomoci soukromého klíče uloženého v kartě
- Nastavování a ověřování pinu

V poslední fázi se data přijatá z čipové karty upraví na podpis podle PKCS#1, který se pak vloží do oblasti připravené ve druhé fázi tak, že **ByteRange** bude zahrnovat celý dokument kromě tohoto podpisu.

6.3 Návrh apletu pro JavaCard

Aplikace na čipové kartě musí umět vytvořit pár klíčů. Soukromý klíč se musí na kartě uchovat a veřejný klíč musí být možné z karty získat, aby mohl být použit v certifikátu podepsující osoby. Aplet bude mít tedy několik funkcí, mezi nimiž se bude volit pomocí instrukčního bitu INS datagramu APDU:

- Vytvoření páru klíčů pro RSA o délce 1024 bitů
- Vytvoření páru klíčů pro RSA o délce 2048 bitů
- Získání veřejného modulu RSA (součást veřejného klíče)
- Získání veřejného exponentu RSA (součást veřejného klíče)
- Dešifrování (podepsání) příchozích dat (hashe) pomocí soukromého klíče uloženého na kartě
- Autentizace uživatele vůči kartě pinem

Hlavní program používá jen autentizační a podepisovací funkci, ostatní funkce se použijí jen jednou před vydáním karty uživateli. Pro používání těchto funkcí a tedy pro účely základní konfigurace karty bude sloužit samostatný program „keymanager“. Tento program bude generovat a posílat kartě datagramy APDU, na jejichž základě karta vytvoří nový klíč nebo odešle zpět veřejný modulus nebo exponent. Tento program také z veřejného modulu vytvoří veřejný klíč ve formátu DER nebo pomocí knihovny OpenSSL vytvoří certifikát. Pomocí veřejného klíče nebo certifikátu se dají ověřovat všechny podpisy, které karta s odpovídajícím soukromým klíčem vytvořila.

Před provedením jakékoliv operace na kartě bude vyžadována autentizace uživatele vůči kartě čtyřmístným číslem PIN. Po pěti nesprávných zadáních tohoto čísla se karta zablokuje. Pro praktické využití by bylo ještě vhodné umožnit odblokování karty jiným tajným číslem, podstatně delším. Dalším způsobem, jak zablokovanou kartu znovu použít je opětovné nahrání apletu poskytovatelem a následné vygenerování nového klíče. Původní klíč bude ale ztracen, takže útočník, který získá ke kartě přístup, nebude moci vytvořit žádný podpis původním klíčem.

Kapitola 7

Bezpečnost

Podívejme se nyní na tento systém z hlediska bezpečnosti. Hrozbou je možnost napadení zařízení, na kterém aplikace běží. Útočník, který by měl možnost ovlivnit chování aplikace, by například mohl kromě naskenovaného dokumentu podepsat i jakákoli jiná data, která by pak mohl zneužít. Aplikace má podle předpokladů běžet na zařízení, které skenuje a podepisuje dokumenty a nemá žádnou jinou funkci. Zařízení je spravováno důvěryhodnou entitou, která ho musí zabezpečit proti případným útokům nebo alespoň zajistit, aby byla neoprávněná manipulace se zařízením odhalena (tamper evidence). Pokud je toto splněno, můžeme zařízení považovat za důvěryhodné.

Další hrozbou je zneužití karty. Útočník by mohl pomocí odcizené karty podepsat libovolný dokument ještě dříve, než by majitel nahlásil ztrátu a došlo by k revokaci certifikátu. Nebo by útočník odcizenou kartu po zneužití opět vrátil nenápadně majiteli, který by se to ani nemusel dozvědět. Proti těmto útokům je karta chráněna pinem. Po pěti neúspěšných pokusech o zadání pinu se aplet na kartě zablokuje a již není možné ho používat. Opětovným nahráním apletu se karta stává znova použitelnou, ale samozřejmě se musí vygenerovat nový pár klíčů, takže původní klíče nejdou zneužít. Útočník má díky pinu nízkou pravděpodobnost úspěchu. Pro získání pinu z klávesnice zařízení útočníkem platí předpoklad o zabezpečení zařízení, který byl rozebrán v předchozím odstavci.

Výše popsané hrozby jsou specifické pro tuto aplikaci, kromě nich mohou být provedeny i jiné útoky, například útok na čipovou kartu pomocí časové nebo odběrové analýzy, což však již není tématem této práce. Další možností je útok na algoritmus RSA. Aplikace podporuje délku klíčů 1024 a 2048 bitů, obě tyto délky jsou v současné době považovány za bezpečné. Doporučuje se používat klíče o délce 2048 bitů, protože budou bezpečné déle. V současné době se dá 512 bitové RSA prolomit s běžným domácím počítačem, 768 bitů je hranice, která se dá prolomit s nejlepší technikou, 1024 bitů je zatím ještě považováno za bezpečnou délku klíče, ale nebude tomu tak příliš dlouho.

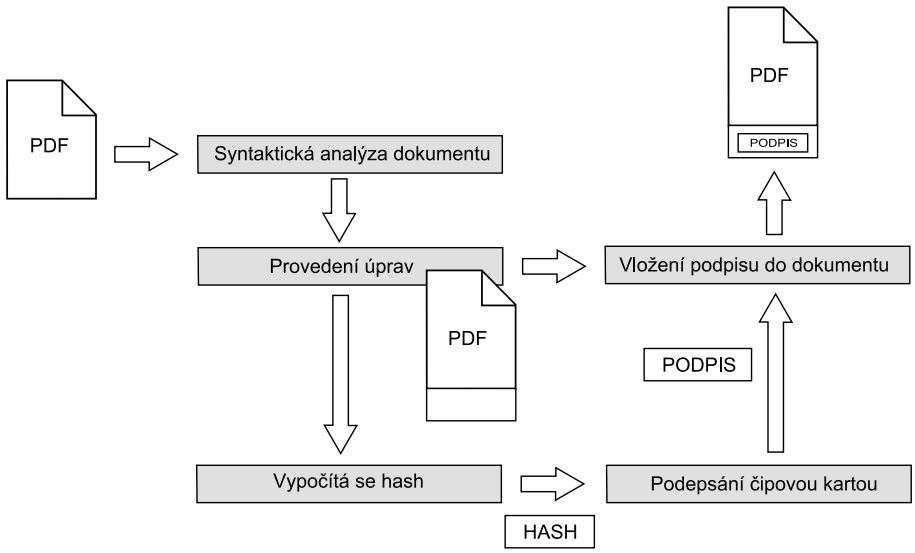
Kapitola 8

Implementace

Pro práci s dokumenty PDF existují různé knihovny, například iText, PJX nebo PDF Box. Všechny volně šiřitelné knihovny jsou však pro jazyk Java. Jelikož mezi požadavky bylo použití co nejmenšího množství externích knihoven a použití programovacího jazyka C++, nejsou tyto knihovny pro tento projekt použitelné. Syntaktická analýza a funkce na vkládání nových objektů musela být tedy naprogramována od začátku. Zdrojové kódy, které obstarávají práci s dokumentem PDF, jsou uloženy v souborech, jejichž názvy začínají písmeny `pdf`. Z těchto zdrojových kódů by se dala vytvořit samostatná knihovna, která by umožňovala syntakticky analyzovat soubory PDF, procházet strukturu a přidávat nebo odebírat objekty na základní úrovni. Jedná se o zdrojové kódy v C++ bez použití dalších knihoven.

Program může být zkompilován překladačem GNU GCC pod operačním systémem Linux. Pro výpočet hashe se využívá kryptografická knihovna OpenSSL, jejíž zdrojové kódy jsou open source. Tato knihovna může být zkompilována pro většinu významných operačních systémů, v našem případě pro Linux. OpenSSL implementuje mimo jiné i kryptografickou hashovací funkci SHA-1. Na komunikaci s čipovou kartou je použita knihovna *pcsclite*, která zprostředkovává přístup k čipové kartě přes PCSC. Pro tuto aplikaci byla použita JavaCard čipová karta GXP E32 PK PRO-R3 v3.2 od firmy Gemplus (nyní Gemalto). Aplet pro kartu je napsán v jazyce Java.

Program funguje v pěti fázích, jak již bylo popsáno. Činnost aplikace je znázorněna na obrázku 8.1.



Obrázek 8.1: Fáze činnosti programu

8.1 Syntaktická analýza dokumentu

Struktura souboru PDF byla popsána ve druhé kapitole. Dokument se skládá z jedné části *Header* a jedné nebo více částí *Body*, *Cross-reference table* a *Trailer*. Část *Body* se skládá z objektů, které mohou být vnořené. Celý dokument je potřeba analyzovat, aby bylo možné přistupovat k objektům, které se budou měnit.

Pro syntaktickou analýzu je vytvořena řada tříd v C++, každá reprezentuje jeden objekt v PDF. Metoda `parse()` každé třídy umí tento objekt v souboru PDF analyzovat. Pokud se jedná o objekt obsahující další objekty, metoda `parse()` vytvoří instance příslušných tříd a zavolá jejich metodu `parse()` a uloží ukazatele na tyto instance. Tímto způsobem je možné syntakticky analyzovat celý dokument a následně v něm vyhledávat a procházet ho pomocí ukazatelů. Všechny třídy jsou potomky třídy `pdfObject`, takže základní funkce dědí z této třídy.

Nejprve se vytvoří instance třídy `pdfFile`, která je také potomkem třídy `pdfObject` a která reprezentuje celý soubor. Metoda `parse()` načte hlavičku souboru a pak vytváří a ukládá ukazatele na části dokumentu, sekce, ze kterých se tento soubor skládá. Těmito sekcemi jsou instance třídy `pdfSection`, které reprezentují jednotlivé skupiny částí *Header*, *Body*, *Cross-reference table* a *Trailer*. Pokud je v souboru jen jedna `pdfSection`, pak se jedná o původní verzi souboru, každá další `pdfSection` znamená, že byla provedena změna pomocí *Incremental update*.

Na konci každé části reprezentované `pdfSection` je vždy uložen řetězec tvaru `startxref\n offset \n %%EOF\n`, kde `offset` značí offset tabulky *cross-reference* a `\n` znak konce řádku. Metoda `parse()` třídy `pdfSection` nejprve najde a zjistí tento offset, podle kterého najde tabulku *cross-reference*. Tuto tabulku pak analyzuje a načte instance třídy `pdfXref`, v jejímž poli pak zůstanou uloženy všechny odkazy na objekty. Dále pak prochází sekci od jejího začátku a vytváří instance tříd podle toho, jaké objekty se v souboru nacházejí, ukládá ukazatele na ně a volá jejich metodu `parse()`.

Pro analýzu základních objektů PDF jsou vytvořeny tyto třídy:

```
pdfArray
pdfBoolean
pdfComment
pdfDictionary
pdfFile
pdfIndirect
pdfIndirecrR
pdfName
pdfNull
pdfNumber
pdfString
```

8.2 Provedení úprav - příprava na vložení digitálního podpisu

Následující dvě třídy pracují s podepisovaným dokumentem. Třída `pdfSigningFile` je potomkem třídy `pdfFile` a jsou do ní přidány metody umožňující digitální podepsání. Třída `pdfSignature` je potomkem třídy `pdfSection` a reprezentuje sekci digitálního podpisu.

Třída `pdfSigningFile` poskytuje metodu `sign()`, která do dokumentu přidá sekci

`pdfSignature` jako *Incremental update*. Metoda `sign()` třídy `pdfSignature` vytváří kompletní podpis. Nejdřív musí provést všechny již popsané změny ve struktuře dokumentu.

Nejprve přidá do kořenového objektu odkaz na *AcroForm*. Ve výsledném dokumentu tak v tomto objektu přibude tento řetězec:

```
/AcroForm %d 0 R
```

%d je *object number* nově vytvořeného objektu *AcroForm*.

Dále se vytvoří objekt *AcroForm*, který bude ve výsledku obsahovat takovýto řetězec:

```
%d 0 obj
<<Fields [%d 0 R ]/SigFlags 3>>
endobj
```

První %d je číslo tohoto objektu *AcroForm*, druhé %d je číslo objektu *Fields*. Atribut **SigFlags** nastavuje příznaky určující chování prohlížeče s ohledem na digitální podpis. Bit 1 značí, že se v dokumentu nachází alespoň jeden digitální podpis, což pro prohlížeč znamená, že upraví uživatelské rozhraní tak, aby se dalo k podpisu přistupovat a prohlížet ho. Bit 2 značí, že dokument obsahuje podpis, který může být zneplatněn. Pokud se dokument uloží, prohlížeč na to musí uživatele před uložením upozornit. Tento program nastavuje první i druhý bit na 1, výsledkem je tedy hodnota 3.

Následuje vytvoření *Field dictionary*, ve kterém musí být pro digitální podpis nastaveny tyto atributy:

- **FT** - Typ pole, který tento slovník popisuje, v případě digitálního podpisu musí být hodnota nastavena na **Sig**
- **Type** - hodnota nastavena na **Annot**
- **Subtype** - hodnota nastavena **Widget**
- **V** - hodnotou je *Indirect reference* na *Signature dictionary*
- **T** - jméno tohoto pole, může být libovolné
- **P** - odkaz *Indirect reference* na objekt reprezentující stránku dokumentu, na které se podpis vyskytuje
- **Rect** - definuje obdélník, ve kterém je podpis zobrazen, jelikož tento program vytváří neviditelný podpis, měla by být velikost obdélníku nastavena na 0
- **AP** - hodnotou je *Dictionary* definující vzhled podpisu na stránce, jelikož se vytváří neviditelný podpis, hodnotou je prázdný slovník
- **H** - *highlighting mode* se nastaví na `N` (no highlighting)
- **F** - *Flags*, u neviditelného podpisu by měl být nastaven bit **Hidden** (bit 2) a **NoView** (bit 6) - tyto bity mohou být nastaveny dekadickým číslem 34.
- **MK** - definuje dodatečné vlastnosti vzhledu, v případě neviditelného podpisu irelevantní
- **BS** - *Border style* - hodnotou je slovník definující styl okraje, u neviditelného podpisu irelevantní

Ve výsledném PDF vypadá tento objekt takto:

```
%d 0 obj
<</FT /Sig/Type /Annot/Subtype /Widget/V %d 0 R/T (Signature1)
/P %d 0 R/Rect [0 0 0 0]/AP <<>>/H /N/F 34/MK <</BG [1 1 1 ]>>
/BS <</S /S/W 0/Type /Border>>>>
endobj
```

Nakonec se vytvoří samotný objekt *Signature dictionary*. Ten obsahuje tyto položky:

- **Cert** - certifikát - hodnotou je buď jeden hexadecimální string, který obsahuje certifikát, nebo pole hexadecimálních stringů, které obsahuje celý certifikační řetěz, v takovém případě je jako první uložen certifikát podepisujícího

- **Content** - hodnota digitálního podpisu (podle PKCS#1), reprezentovaná hexadecimálním stringem, nejprve se řetězec ponechá prázdný, hodnota se vkádá až po vypočtení podpisu ze zbytku dokumentu

- **ByteRange** - tato položka určuje intervaly v dokumentu, které se podepisují - 1. a 3. hodnota pole určují na jakém offsetu interval začíná, 2. a 4. hodnota určují délku daného intervalu

- **SubFilter** - specifikuje, jak je hodnota digitálního podpisu v dokumentu kódována - pro kódování PKCS#1 se použije hodnota **adbe.x509.rsa_sha1**

- **Filter** - jméno preferovaného *Signature handleru*

- **Type** - pokud je přítomen, měl by být nastaven na **Sig**

První hodnota pole v položce `ByteRange` je offset začátku, tedy 0. Druhá hodnota je délka prvního intervalu, který končí za znakem „<“ položky `Content`. Dá se snadno spočítat, protože veškerý obsah dokumentu před tímto znakem je již znám. Třetí hodnota je offset začátku druhého intervalu, který začíná před znakem „>“ položky `Content`. Rozdíl mezi třetí a druhou hodnotou odpovídá délce podpisu, která je v případě PKCS#1 dopředu známa (protože neobsahuje certifikát, ale jen hodnotu podpisu kódovanou pomocí ASN.1). Čtvrtá hodnota pole definuje délku druhého intervalu. Tato hodnota se zatím nemůže vyplnit (nastaví se na nuly), protože délka zbytku dokumentu ještě není známa, protože se ještě musí vytvořit tabulka *Cross-reference* a *Trailer*. Tuto hodnou program doplní hned jak je to možné.

Signature dictionary bude v této fázi vypadat takto:
/Cert <****certificate****>/Content <>/ByteRange [%d %d %d 0000000000]
/SubFilter /adbe.x509.rsa_sha1/Filter /Adobe.PPKLite/Type /Sig>>
endobj
pozn. položka Cert bude obsahovat certifikát v hexadecimální formě, položka Content zatím zůstává prázdná

Zbývá vytvořit tabulku *Cross-reference* a *Trailer*. Po jejich vytvoření a vložení je známa délka nového dokumentu, takže se doplní poslední položka do `ByteRange`.

8.3 Vypočítání hashe požadované části připraveného dokumentu

Metoda `sign()` třídy `pdfSignature` po dokončení úprav volá metodu s názvem `createSignature()`, která je odpovědná za vypočítání digitálního podpisu z dat v definované oblasti. Používá k tomu metody třídy `scSign`, což je třída nezávislá na dokumentu PDF, která provádí kryptografické operace nad daty, které jsou jí poslány. Třída `scSign` provádí hashovací funkci na počítači, implementována je SHA-1. RSA provádí tato třída pomocí čipové karty. Pro testovací účely se dá podepisovat i na počítači bez použití čipové karty a to algoritmem RSA nebo DSA.

8.4 Podepsání hashe čipovou kartou

O podepisování na čipové kartě se stará metoda `signOnSc()`. Nejprve se připojí k čipové kartě, autentizuje se číslem PIN, potom pošle data, která se mají dešifrovat algoritmem RSA. Protože je maximální velikost dat v APDU datagramu posílaního na kartu jen 255 bajtů, tak se do něj nevejde celý blok, který má v případě 2048 bitového RSA 256 bajtů. Z tohoto důvodu se padding neprovádí na PC, ale až na čipové kartě. Na čipovou kartu se posílají jen nezarovnaná data, konkrétně identifikace hashovacího algoritmu SHA-1 v kódování ASN.1, která má 15 bajtů, následovaná 20 bajty obsahující hodnotu hash. Dohromady se tedy posílá 35 bajtů v případě RSA 1024 i RSA 2048. Na kartě se potom provede zarovnání těchto 35 bajtů do bloku potřebné velikosti. Maximální velikost dat v APDU datagramu posílaného z karty do PC je 256 bajtů, takže je možné poslat naráz celý blok i v případě RSA 2048.

8.5 Vložení digitálního podpisu do připraveného dokumentu

Na závěr se vloží hodnota digitálního podpisu mezi znaky „<“ a „>“ položky `Content` objektu *Signature dictionary*. Do výstupního souboru se pak uloží celý vstupní soubor, za kterým následuje nová sekce obsahující úpravy a digitální podpis. Výsledkem je digitálně podepsaný PDF dokument.

Každý PDF prohlížeč, který tuto funkci nabízí, může platnost tohoto podpisu ověřit, certifikát (případně řetězec certifikátů) potřebný k ověření je v dokumentu obsažen. Integrita dokumentu se dá ověřit kdykoliv, stačí ověřit, že podpis byl vytvořen soukromým klíčem, který odpovídá veřejnému klíči uloženému v certifikátu podepisujícího. Pokud je podepisující certifikát důvěryhodný, tak můžeme ověřit také autenticitu a prokázat nepopiratelnost.

8.6 Aplet pro JavaCard

Pro vývoj apletu byl použit Borland JBuilder7, GemXpresso RADIII a JavaCard Kit 2.2.2. Aplet implementuje všechny funkce popsané v návrhu, je tedy možné vytvořit pár klíčů a získat veřejný klíč z karty. Dále je možné dešifrovat (podepsat) data poslaná na kartu.

O generování klíčů, podepisování dat a přenos APDU datagramů se stará JavaCard API.

Vytvoření páru klíčů

Pro RSA s délkou klíče 2048 bitů se používá soukromý klíč ve variantě CRT (využívá Čínskou větu o zbytku, angl. Chinese remainder theorem).

Následující příkazy vytvoří objekty klíčů:
```
m_privateKey = KeyBuilder.buildKey(KeyBuilder.TYPE_RSA_CRT_PRIVATE,
  KeyBuilder.LENGTH_RSA_2048,false);
m_publicKey = KeyBuilder.buildKey(KeyBuilder.TYPE_RSA_PUBLIC,
  KeyBuilder.LENGTH_RSA_2048,true);
m_keyPair = new KeyPair(KeyPair.ALG_RSA_CRT,
  (short) m_publicKey.getSize());
```

Klíče se vygenerují tímto příkazem:
```
m_keyPair.genKeyPair();
```

Získání ukazatelů na klíče:
```
m_publicKey = m_keyPair.getPublic();
m_privateKey = m_keyPair.getPrivate();
```

Vytvoření instance šifrovacího algoritmu:
```
m_decryptCipher = Cipher.getInstance(Cipher.ALG_RSA_NOPAD, false);
m_decryptCipher.init(m_privateKey, Cipher.MODE_DECRYPT);
```

Veřejný modulus se získá a odešle v APDU datagramu takto:
```
((RSAPublicKey) (m_publicKey)).getModulus(apdubuf, (short) 0);
apdu.setOutgoingAndSend((short) 0,
  (short)(m_publicKey.getSize() / 8));
```

Podepsání dat
Jelikož se na kartu neposílá celý zarovnaný blok, ale jen identifikace hash funkce a hodnota hash, je nutné data zarovnat (padding). Zarovnání do bloku podle PKCS#1 vypadá tak, že první bajt je 0x00, druhý bajt 0x01, potom hodnoty 0xFF, za nimi jeden bajt 0x00 následovaný zarovnávanými daty, která končí na konci bloku. Počet hodnot 0xFF závisí na délce bloku a na délce zarovnávaných dat.

Zarovnání provádí následující kód:
```
Util.arrayFillNonAtomic(m_padded,(short)0,(short)260,(byte)0xFF);
m_padded[0] = (byte) 0x00;
m_padded[1] = (byte) 0x01;
//len je počet bajtů bloku, tedy 256 pro RSA 2048
//dataLen je délka zarovnávaných dat, v případě SHA-1 je to 35
m_padded[(short)(((short)len-1)-(short)dataLen)] = (byte) 0x00;
Util.arrayCopyNonAtomic(apdubuf, ISO7816.OFFSET_CDATA, m_padded,
  (short) (len-dataLen),(short) dataLen);
```

Inicializace šifrovacího algoritmu:
```
m_decryptCipher.init(m_privateKey, Cipher.MODE_DECRYPT);
```

Dešifrování bloku algoritmem RSA:
```
len = m_decryptCipher.doFinal(m_padded, (short) 0, (short) len,
  m_ramArray, (short) 0);
```

Zkopírování bloku a odeslání pomocí APDU datagramu:

```
Util.arrayCopyNonAtomic(m_ramArray, (short) 0, apdubuf, (short) 0,
   (short) len);
apdu.setOutgoingAndSend((short) 0, (short) len);
```

Kapitola 9

Závěr

Navržený systém je funkční a z hlediska bezpečnosti pro dané účely vyhovující. Byl testován na PC pod operačním systémem Linux (Ubuntu 8.10 a openSUSE 11.1). Cílové zařízení nebylo až do odevzdání práce k dispozici, takže na něm nemohl být program otestován, s instalací na toto zařízení by však neměl být problém. Aplikace podporuje verze PDF 1.0 až 1.4, novější verze jsou komplikovanější na rozbor. Vzhledem k tomu, že se dokumenty vytváří přímo na skeneru, neměl by být problém je vytvořit v potřebné verzi. Podepsané dokumenty byly testovány pod Windows v Adobe Reader ver. 7 a 8, dále pod Linuxem v Adobe Reader 9, zobrazení dokumentu, podpisu i certifikátu bylo v pořádku.

Podpis se vkládá podle standardu PKCS#1 v1.5. Pro podpis se podle tohoto standardu používá hashovací algoritmus SHA-1 a šifrovací algoritmus RSA. Aplikace podporuje velikosti bloků 1024 a 2048 bitů. Pro testovací účely je možné podepisovat i bez pomoci čipové karty. K tomu musí být nainstalována knihovna OpenSSL. Vytvořený podpis pak odpovídá standardu PKCS#7.

Čipová karta je proti zneužití chráněna pinem. Po pěti nesprávných pokusech o autentizaci uživatele vůči kartě se karta zablokuje a je nutné nahrát aplet znovu a vygenerovat nový klíč.

Kromě hlavního programu *pdfsign*, který slouží k podepisování PDF dokumentů, byl vytvořen program *keymanager*, který slouží ke generování klíčů a získávání veřejného modulu a exponentu, dále k podepisování bloku dat a k nastavování a verifikaci pinu.

Komunikace mezi programem a kartou není šifrována, protože se nepřenáší žádná citlivá data, která by mohla být odposlechnuta a zneužita. Jediným nebezpečím je podvržení komunikace, kdy by se kartě poslala podvržená data, která by byla kartou podepsána v domnění, že se jedná o hash pravého dokumentu. Útočník by mohl například poslat hash jiného dokumentu, který by si tímto způsobem mohl nechat podepsat napadeným uživatelem bez jeho vědomí. Při odcizení karty je toto nebezpečí minimální, protože karta je chráněna pinem. Další možností je ale škodlivý software, který běží na zařízení a který po zadání pinu uživatelem pošle na kartu podvržený hash k podepsání. Jelikož je zařízení určeno pouze ke skenování a podepisování dokumentů, je k němu omezený přístup a je obtížné škodlivý software nainstalovat. I tak by ale mělo být proti těmto útokům chráněno minimálně tak, aby byl každý pokus o nedovolenou manipulaci se zařízením odhalen (tamper evidence).

Programovou část projektu tvoří podepisovací program, program na správu karty a aplet pro čipovou kartu. Zdrojové soubory, jejichž názvy začínají písmeny `pdf`, slouží k analýze dokumentu a k základním operacím nad strukturou souboru PDF, jako je procházení struktury dokumentu, vkládání a mazání objektů nebo provádění *incremental update*. Tyto zdrojové kódy by se daly použít i v jiných aplikacích zaměřených na práci s PDF, případně by se mohly více rozvinout a dala by se z nich vytvořit samostatná knihovna.

Literatura

[1] ISO 32000-1:2008
`http://www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf`

[2] [online] Adobe, PDF Reference and Adobe Extensions to the PDF Specification
`http://www.adobe.com/devnet/pdf/pdf_reference.html`
[cit. 2009-5-22]

[3] [online] PDF Tools AG, PDF Primer
`http://www.pdf-tools.com/public/downloads/whitepapers/whitepaper-pdfprimer.pdf`
[cit. 2009-5-22]

[4] [online] Adobe, Quick overview of PDF file format
`http://www.adobe.com/devnet/livecycle/articles/lc_pdf_overview_format.pdf`
[cit. 2009-5-22]

[5] Zákon č. 227/2000 Sb., o elektronickém podpisu
`http://www.mvcr.cz/clanek/zakon-c-227-2000-sb-o-elektronickem-podpisu.aspx`

[6] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone
Hanbook of applied cryptography
CRC Press, ISBN: 0-8493-8523-7, October 1996, 816 pages
Chapter 11 - Digital Signatures
`http://www.cacr.math.uwaterloo.ca/hac/`

[7] RSA Laboratories, PKCS#1 Version 1.5
`ftp://ftp.rsasecurity.com/pub/pkcs/ps/pkcs-1.ps`

[8] RSA Laboratories, PKCS#7
`ftp://ftp.rsasecurity.com/pub/pkcs/ps/pkcs-7.ps`

[9] [online] Denis Royer, A study on PKI and biometrics
`http://www.fidis.net/resources/deliverables/hightechid/int-d32000/doc/7/`
[cit. 2009-5-22]

[10] [online] CardWerk, The ISO 7816 Smart Card Standard
http://www.cardwerk.com/smartcards/
smartcard_standard_ISO7816.aspx
[cit. 2009-5-22]

[11] [online] Gemalto, Developer network - JavaCard
http://developer.gemalto.com/home/java-card.html
[cit. 2009-5-22]

Příloha A

Minimální PDF dokument

Tato příloha ukazuje vzor minimálního dokumentu PDF, jak je popsán v ISO 32000-1:2008. Tento dokument obsahuje jednu stránku s nápisem „Hello World“.

```
%PDF-1.4
1 0 obj
<< /Type /Catalog
/Outlines 2 0 R
/Pages 3 0 R
>>
endobj
2 0 obj
<< /Type /Outlines
/Count 0
>>
endobj
3 0 obj
<< /Type /Pages
/Kids [4 0 R]
/Count 1
>>
endobj
4 0 obj
<< /Type /Page
/Parent 3 0 R
/MediaBox [0 0 612 792]
/Contents 5 0 R
/Resources << /ProcSet 6 0 R
/Font << /F1 7 0 R >>
>>
>>
endobj
5 0 obj
<< /Length 73 >>
stream
BT
/F1 24 Tf
100 100 Td
(Hello World) Tj
```

```
ET
endstream
endobj
6 0 obj
[/PDF /Text]
endobj
7 0 obj
<< /Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Helvetica
/Encoding /MacRomanEncoding
>>
endobj
xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000177 00000 n
0000000318 00000 n
0000000411 00000 n
0000000439 00000 n
trailer
<< /Size 8
/Root 1 0 R
>>
startxref
547
%%EOF
```

Příloha B

Digitálně podepsaný minimální PDF dokument (PKCS#7)

Tato příloha obsahuje digitálně podepsaný minimální PDF dokument s nápisem „Hello World“. Podpis je vytvořen podle normy PKCS#7. Byl vytvořen na PC tímto programem bez čipové karty.

```
%PDF-1.4
1 0 obj
<< /Type /Catalog
/Outlines 2 0 R
/Pages 3 0 R
>>
endobj
2 0 obj
<< /Type /Outlines
/Count 0
>>
endobj
3 0 obj
<< /Type /Pages
/Kids [4 0 R]
/Count 1
>>
endobj
4 0 obj
<< /Type /Page
/Parent 3 0 R
/MediaBox [0 0 612 792]
/Contents 5 0 R
/Resources << /ProcSet 6 0 R
/Font << /F1 7 0 R >>
>>
>>
endobj
5 0 obj
<< /Length 73 >>
stream
BT
/F1 24 Tf
100 100 Td
```

```
(Hello World) Tj
ET
endstream
endobj
6 0 obj
[/PDF /Text]
endobj
7 0 obj
<< /Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Helvetica
/Encoding /MacRomanEncoding
>>
endobj
xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000177 00000 n
0000000318 00000 n
0000000411 00000 n
0000000439 00000 n
trailer
<< /Size 8
/Root 1 0 R
>>
startxref
547
%%EOF
1 0 obj
<< /Type /Catalog
/Outlines 2 0 R
/Pages 3 0 R
/AcroForm 8 0 R

>>
endobj

8 0 obj
<</Fields [9 0 R ]/SigFlags 3>>
endobj
9 0 obj
<</FT /Sig/Type /Annot/Subtype /Widget/V 10 0 R/T (Signature1)
   /P 4 0 R/Rect [0 1 0 1]/AP <<>>/H /N/F 4/MK <</BG [1 1 1 ]>>
```

```
   /BS <</S /S/W 0/Type /Border>>>>
endobj
10 0 obj
<</Contents <3082067406092a864886f70d010702a08206653082066102010
1310b300906052b0e03021a0500300b06092a864886f70d010701a0820481308
2047d30820365a003020102020900f99220edb138fab6300d06092a864886f70
d0101050500308185310b3009060355040613024135a31173015060355040813
e437a65636820526570756 26c6963310d300b060355040713044726e6f311b3
019060355040a13124d61736172796b20556e697665727369747931 1f301d060
355040b1316466163756c7479206f6620496e666f726d617469633110300e0
603550403130754455354204341301e170d3039303531323233343830365a170
d3132303531313233343830365a308185310b3009060355040613024135a31173
0150603550408130e437a65636820526570756 26c6963310d300b06035504071
30442726e6f311b3019060355040a13124d61736172796b20556e69766572736
97479311f301d060355040b1316466163756c7479206f6620496e666f726d617
469633110300e060355040313075445535420434130820122300d06092a864
886f70d01010105000382010f003082010a0282010100b13df49283bf7dae17e
dfc6f64f71fdbc2d232a43d743211fefeeebdc3d842e9ca973adec9e5e3790ef
aa04a303c3a2b93234beeec2582f5a3facc49ecb285a586e499ce390b5273e83
7e9ca51d033f2ecd6abd2fbc3a1de889b7b10265efa71a8f6f684741e7419c59
ca3c1bae4418d76af312b965c7fae7fd70b11c47caa4c2c0b31117c4d4c8299d
8d1af9fa76d6bf9f1c8c86a7d0079f134d6ab95af60368be97962305b19b7534
17c31dab0b66a26939f68d619a7f993b1e241d8f37ab4451e6027d69c54810af
0bf6241eea78cb0ef82659a7c27b6f57dd4ed8cb1ae74ebd3e5b437c953892ed
c88408b7afdb38887489bb0563a2d432cb32fec2c4cc10203010001a381ed308
1ea301d0603551d0e0416041405c08635b6fe44840006ba8a0c2afa74996cd1d
83081ba0603551d230481b23081af801405c08635b6fe44840006ba8a0c2afa7
4996cd1d8a1818ba4818830818531 0b3009060355040613024135a31173015060
3550408130e437a65636820526570756 26c6963310d300b06035504071304427
26e6f311b3019060355040a13124d61736172796b20556e69766572736974793
11f301d060355040b1316466163756c7479206f6620496e666f726d617469637
33110300e060355040313075445535420434182090 0f99220edb138fab6300c0
603551d13040530030101ff300d06092a864886f70d0101050500038201010 04
5643b63a41aa01aea3b2bb915c51780724cac1a92eb9cb87c40785f8ebaf5125
b688f366359a28dacd48a04d814655de68966cb92baf0a9c05f5d99cbce89cde
db2854d6dbadb0e5371f66debb8fa07de9ce4d717c2c7a733a13b311ebeb2858
15930e9216e19184bfbe0127b5139476508bee8c7493849eb0fcd2db61d792f2
970106a08f8e932e6f097b1e7671064cd3d0f32eaf3fa4d46b75ef62244baa50
fb1e9a91bd0dc7915798bf70dcbc038a92615a8c8382eca44d9c1be675353682
c7659c5a40a5df56c96c3043bead14c575832f807da4283e01420bb540b42ce3
5f48fce26242aecefe035445ca6650e39340b85e01d64ad07b02d9706e88c963
18201bb308201b702010130819330818531 0b3009060355040613024135a31173
0150603550408130e437a65636820526570756 26c6963310d300b06035504071
30442726e6f311b3019060355040a13124d61736172796b20556e69766572736
97479311f301d060355040b1316466163756c7479206f6620496e666f726d617
469633110300e060355040313075445535420434102090 0f99220edb138fab
6300906052b0e03021a0500300d06092a864886f70d0101010500048201004 85
```

```
e8a575abd23349d6623694a8bad854fd9fe7dc9f33d94f0b7c41dc0197ee5708
f4a992d07eb9ea6cd69cd8aa27121f2096482aa229d651882ac4e6018e1866e1
76ddd2cffc8f1e51795a60351bd5b4b1872e6af4ebc8270353181cb887cdcd7c
1cda643bdfabbc10097cb49a4b87c7a1161da8b5494504b4509599490434bb93
000ec056737280434a4f85f63f6bfd19bda64a831ff79948eb72da93344c233a
7da88183b8251c5d22bcdfc4d5566928b65a8f9b9dc567960a3dacbda5701f98
e7fc70ecf329d9721c32c59c44250b0873a12fb9abe8d52f7cd57d4bf89a8b01
e78da80597caaa93f0739638ad254f6d18019110bd2d2dc7676dc37e2d42b000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000>/ByteRange [000000
0000 0000001086 0000005088 0000000400]/Location (CZ)/Reason (Doc
ument Seal)/M (D:20090518212512+02’00’)/SubFilter /adbe.pkcs7.de
tached/Filter /Adobe.PPKLite/Type /Sig>>
endobj
xref
0 1
0000000000 65535 f
1 1
0000000762 00000 n
8 1
0000000844 00000 n
9 1
0000000892 00000 n
10 1
0000001062 00000 n
trailer
<<
/Size 12
/Root 1 0 R
/Prev 547
>>
startxref
5282
%%EOF
```

Příloha C

Digitálně podepsaný minimální PDF dokument (PKCS#1)

Tato příloha obsahuje digitálně podepsaný minimální PDF dokument s nápisem „Hello World“. Podpis je vytvořen podle normy PKCS#1 v1.5. Byl vytvořen vytvořen algoritmem RSA s délkou klíče 2048 bitů pomocí čipové karty.

```
%PDF-1.4
1 0 obj
<< /Type /Catalog
/Outlines 2 0 R
/Pages 3 0 R
>>
endobj
2 0 obj
<< /Type /Outlines
/Count 0
>>
endobj
3 0 obj
<< /Type /Pages
/Kids [4 0 R]
/Count 1
>>
endobj
4 0 obj
<< /Type /Page
/Parent 3 0 R
/MediaBox [0 0 612 792]
/Contents 5 0 R
/Resources << /ProcSet 6 0 R
/Font << /F1 7 0 R >>
>>
>>
endobj
5 0 obj
<< /Length 73 >>
stream
BT
/F1 24 Tf
100 100 Td
```

```
(Hello World) Tj
ET
endstream
endobj
6 0 obj
[/PDF /Text]
endobj
7 0 obj
<< /Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Helvetica
/Encoding /MacRomanEncoding
>>
endobj
xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000177 00000 n
0000000318 00000 n
0000000411 00000 n
0000000439 00000 n
trailer
<< /Size 8
/Root 1 0 R
>>
startxref
547
%%EOF
1 0 obj
<< /Type /Catalog
/Outlines 2 0 R
/Pages 3 0 R
/AcroForm 8 0 R

>>
endobj

8 0 obj
<</Fields [9 0 R ]/SigFlags 3>>
endobj
9 0 obj
<</FT /Sig/Type /Annot/Subtype /Widget/V 10 0 R/T (Signature1)
   /P 4 0 R/Rect [0 1 0 1]/AP <<>>/H /N/F 4/MK <</BG [1 1 1 ]>>
```

```
   /BS <</S /S/W 0/Type /Border>>>>
endobj
10 0 obj
<</Contents <04820100a16c4dee9a6c6c60baf225836ad9ba28a399573ae21
1bdb10319dbd01545561e0e56122184048e82f753d62efb96c3f67873c830260
ced1cd02fecccd8b1f3ff2d36a0b0eaf47d223a4353c4d1a50b3f3609218d676
75058a65b71f35c4de102f431fc64121ec314eb32d0cd4320112c2ddcf72e7f3
67a41344af1196fed496d510abac648f93e91fa9731b1b7826d450e74766541c
f9fa79a578dff5a56aaf537296ab8446fd326cbe808f7fec2045ddf4adde448e
bace7f24e5d7a579f4f9c00c4b847d86ae79a46975548b367ae2ff35c054fda9
47bc9cc626c6bf6bc595e5608a1781e27937d84664d8552da7ef9b2956d7675f
0d07f22dca25988f302c2>/ByteRange [0000000000 0000001086 00000016
08 0000002215]/Location (CZ)/Reason (Document Seal)/M (D:2009051
8212437+02’00’)/SubFilter /adbe.x509.rsa_sha1/Cert <30820384308
2026c020101300d06092a864886f70d0101050500308185310b3009060355040
61302435a311730150603550408130e437a65636820526570756 26c6963310d3
00b060355040713044272 6e6f311b3019060355040a13124d61736172796b205
56e697665727369747931 1f301d060355040b1316466163756c7479206f66204
96e666f726d6174696373311030 0e060355040313075445535420434 1301e170
d303930353133313833313430 5a170d313230353132313833313430 5a3081893
10b300906035504061302435a311730150603550408130e437a6563682052657
075626c6963310d300b060355040713044272 6e6f310b3009060355040a13024
d55310b3009060355040b13024649311530130603550403130c4d617274696e2
048656e7a6c3121301f06092a864886f70d0109011612393835393440 6d61696
c2e6d756e692e637a30820122300d06092a864886f70d0101010500038201 0f0
03082010a0282010100cea1b3010a856ec10b8db9a00be7136aad23d7decd24b
cfd81a266572d634337b95281e507fae531bbb7eb418128bc57b31af59a3b91a
40392c55f212d8458e42671ba69e459e011489d11b0471a2c51a6d1a6ec9d7ab
02ccaf9e527043623bb59afc66762894b3029619b0a4046cc4bbf3740cda720d
9803df34223155daf1b5632ae0abaf54a1a09260c3995bb12b6f1db96d5c3c37
54363d9acf0716900d1864131c222619dc43c21a91255434734b4dcd37482df5
1833bcdae94224b296a7666a76f1280171306bdd9bac5cf95058ede362ecf6da
3867043db4ac5b9a0f3c36161de3f19dbb37ae969d3d0ac2c0e4ca4b6da8490d
3de675c42213dd1ec1b0203010001300d06092a864886f70d010105050003820
101006fb8965dc6c1607b4c84d5ef43c4dc70c44744ef6a2b2fe01bb13e6f2df
2d0e35a23981fcbcba7df3d753bbfc46eeb0bceec10e4af7c5b79af76e6dd527
da872ae7b302a5d86aa5e2cfc041c4e788216d3c469b47506299ac71449d9e74
5d7bd86b8a9f62cb70a92f194c8d60b8c0db9e7823841c9810c729370c72c562
4347b6f08e55d73b378da331907abf08c00a799c410381f8b660a8f5abd5ee8a
a80a557573e1d30e5a645739469a9b8d61b7e1027838f0af52292fba6a8971dd
c35e9f3965047488cfcafa8a117ec4559932713b1f0bf1f81e800ae661de7a1c
0a8e548d050edcfc462f39d4ee0668adb077a8743b93cf9871dbc7313b455b92
4e266>/Filter /Adobe.PPKLite/Type /Sig>>
endobj
xref
0 1
0000000000 65535 f
```

```
1 1
0000000762 00000 n
8 1
0000000844 00000 n
9 1
0000000892 00000 n
10 1
0000001062 00000 n
trailer
<<
/Size 12
/Root 1 0 R
/Prev 547
>>
startxref
3617
%%EOF
```

Příloha D

Ukázka komunikace s čipovou kartou

Zařízení komunikuje s čipovou kartou pomocí APDU datagramů. Toto je ukázka komunikace při podepisování dokumentu pomocí algoritmu RSA s délkou klíče 2048 bitů.

Výběr apletu (SELECT):

```
Sending: 00 A4 04 00 0A 52 53 41 41 70 70 6C 65 74 01 00
Received: 90 00
```

Autentizace uživatele vůči kartě pomocí pinu:

```
Sending: B0 56 00 00 04 30 30 30 30 00
Received: 90 00
```

Odeslání identifikace algoritmu SHA-1 a hodnoty hashe SHA-1. Následuje přijetí podepsaného (dešifrovaného algoritmem RSA) bloku o velikosti 256 bajtů:

```
Sending: B0 52 00 00 23 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04
14 63 A3 34 CA F1 D8 CB 4B 03 99 C1 88 D6 E5 78 3F 12 4B 1E 29 00
Received: A1 6C 4D EE 9A 6C 6C 60 BA F2 25 83 6A D9 BA 28 A3 99
57 3A E2 11 BD B1 03 19 DB D0 15 45 56 1E 0E 56 12 21 84 04 8E 82
F7 53 D6 2E FB 96 C3 F6 78 73 C8 30 26 0C ED 1C D0 2F EC CC D8 B1
F3 FF 2D 36 A0 B0 EA F4 7D 22 3A 43 53 C4 D1 A5 0B 3F 36 09 21 8D
67 67 50 58 A6 5B 71 F3 5C 4D E1 02 F4 31 FC 64 12 1E C3 14 EB 32
D0 CD 43 20 11 2C 2D DC F7 2E 7F 36 7A 41 34 4A F1 19 6F ED 49 6D
51 0A BA C6 48 F9 3E 91 FA 97 31 B1 B7 82 6D 45 0E 74 76 65 41 CF
9F A7 9A 57 8D FF 5A 56 AA F5 37 29 6A B8 44 6F D3 26 CB E8 08 F7
FE C2 04 5D DF 4A DD E4 48 EB AC E7 F2 4E 5D 7A 57 9F 4F 9C 00 C4
B8 47 D8 6A E7 9A 46 97 55 48 B3 67 AE 2F F3 5C 05 4F DA 94 7B C9
CC 62 6C 6B F6 BC 59 5E 56 08 A1 78 1E 27 93 7D 84 66 4D 85 52 DA
7E F9 B2 95 6D 76 75 F0 D0 7F 22 DC A2 59 88 F3 02 C2 90 00
```