PKI a elektronický podpis

Libor Dostálek, 2020

Obsah

1 Základní kryptografické algoritmy

1.1 Otisk (hash)

Nejprve si ukážeme, jak mocným nástrojem je otisk (*hash*). Otisk (Obrázek 1.1) je jednocestná funkce, která nám z libovolně dlouhého textu vytvoří krátký řetězec konstantní délky. Výsledný řetězec (otisk) by měl maximálně charakterizovat původní text. Algoritmy pro výpočet otisku nejsou v žádném případě šifrovacími algoritmy (už vzhledem k nejednoznačnosti – obecně neexistuje inverzní funkce), ale používají se v roli kvalitního "otisku dat".

Tabulka 1.1 Algoritmy tříd SHA

	Algoritmus orisku	Výstup v bitech	Publikován	Aktuální standard
SHA-1		160	1995	FIPS 180-4
SHA-2	SHA-224	225	2001	FIPS 180-4
	SHA-256	256		FIPS 180-4
	SHA-384	384		FIPS 180-4
	SHA-512	512		FIPS 180-4
	SHA-512/224	224		FIPS 180-4
	SHA-512/256	256		FIPS 180-4
SHA-3	SHA3-224	224	2015	FIPS 202
	SHA3-256	256		FIPS 202
	SHA3-384	384		FIPS 202
	SHA3-512	512		FIPS 202

Jednocestnou funkcí se rozumí algoritmy transformující právu, které nejsou výpočetně náročné. Je však výpočetně velice náročné k výsledku nalézt původní text zprávy. Jednocestnou funkci lze přirovnat k manželství. Je přece jednoduché se oženit, ale často velice obtížné se rozvést.

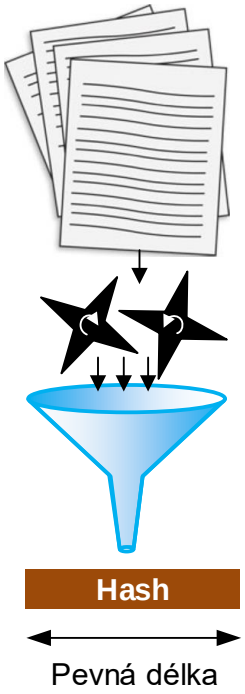
Kvalitní jednocestné funkce pro výpočet otisku by měly dát výrazně jiný výsledek při drobné změně původního textu i při vložení dalšího textu.

Jelikož se otisk počítá z libovolně dlouhého textu, tak ke konkrétnímu otisku je teoreticky možné najít nekonečně mnoho původních textů. U některých zastaralých algoritmů se již daří nacházet texty se stejným otiskem. Výsledkem je pak opouštění těchto algoritmů a jejich nahrazení jinými (např. algoritmy třídy SHA-2, SHA-3 apod.).

Problém je v případě velmi krátkých zpráv, kdy text původní zprávy lze snadno nalézt hrubou silou.

Jednocestné funkce jsou konstruovány na výpočetních operacích nízké úrovně (především bitové operace a posuny) a jsou tedy výpočetně velmi rychlé a efektivní.

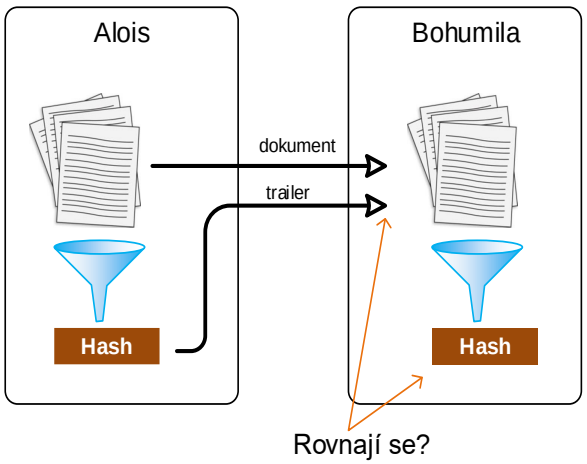
Jak využije Alois otisk ve své komunikaci se svou milou Bohumilou? No přece využije otisk jako důkaz, že zpráva na cestě od Aloise k jeho milé Bohumile nebyla změněna, tj. využije otisk jako důkaz zachování integrity zprávy. Alois neodešle pouze samotná data (text) zprávy, ale data doplní o patu zprávy (*trailer*) obsahující otisk z textu zprávy (Obrázek 1.2).



Obrázek 1.1 Hash

Bohumila, poté co přijme zprávu, spočte otisk z přijaté zprávy a porovná svůj výsledek s otiskem ze zápatí přijaté zprávy (tj. s otiskem spočteným Aloisem). Pokud jsou oba otisky shodné, pak zpráva nebyla cestou změněna. Bohumila ověřila integrity zprávy. Tento typ důkazu zachování integrity přenášených dat využívají např. síťové protokoly pro detekci chyb vzniklých poruchami linek (poruchami fyzické vrstvy komunikace v počítačové síti). Avšak tyto protokoly používají velice jednoduché algoritmy, takže se hovoří o kontrolním součtu a nikoliv otisku.

Algoritmus výpočtu otisku používaný Aloisem a Bohumilou je veřejně popsán v technické normě. Tuto normu si přečte i žárlivý Cyril, který zprávu od Aloise pozmění v jeho neprospěch a do výsledku spočte otisk z pozměněné zprávy (Obrázek 1.3).

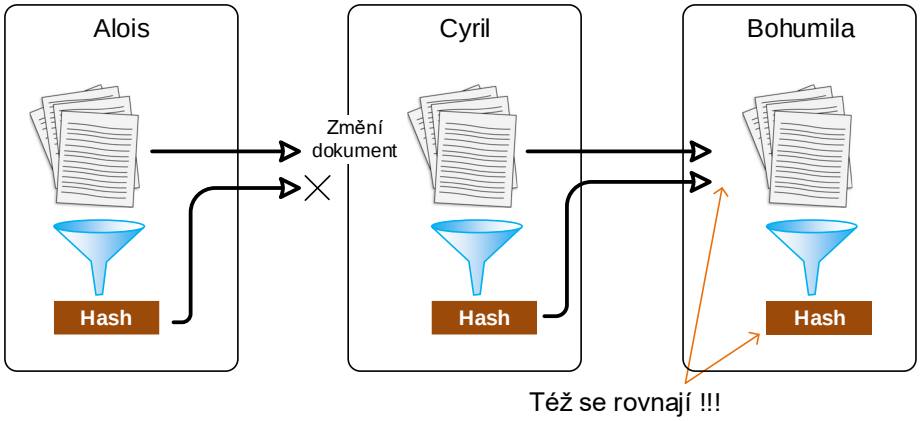


Obrázek 1.2 Útok na integritu zprávy zajištěnou na bázi otisku

1.1.1 HMAC

Naštěstí Alois s Bohumilou Cyrila znají, proto si při své tajné schůzce vymění tajemství (šipka 1 na Obrázek 1.4), které sdílí pouze Alois s Bohumilou a Cyril je tudíž nezná. Jako sdílené tajemství stačí např. nějaký krátký textový řetězec.

Od chvíle, kdy si Alois s Bohumilou vyměnili sdílené tajemství, tak vždy, když bude Alois odesílat nějakou zprávu Bohumile, tak otisk nebude počítat pouze ze zprávy, ale vždy do výpočtu otisku, předem dohodnutým způsobem, zahrne i sdílené tajemství (Obrázek 1.4). Jelikož Cyril tajemství nezná, tak není schopen takovýto otisk spočíst, proto nemůže pozměňovat zprávu, aniž by to Bohumila nepoznala.



Obrázek 1.3 Žárlivý Cyril útočí

Otisk spočtený nejenom ze zprávy, ale ze zprávy nějakým způsobem zřetězené se sdíleným tajemstvím se často označuje jako algoritmus na ochranu integrity na bázi otisku - zkratkou HMAC (*Hash-based Message Authentication Code*). Způsob, jak počítat HMAC byl popsán v RFC 2104, které vyšlo v roce 1997. Tento standard je obecně aplikovatelný na různé algoritmy pro výpočet otisku, takže v praxi pak používáme algoritmy HMAC-MD5, HMAC-SHA1, HMAC-256 apod.

HMAC se využívá velice často. Např. v protokolech SSL/TLS, protokolu IPsec, ale mnohdy jsou na této technice postaveny i autentizační kalkulátory pro tvorbu jednorázových hesel.

Důležité je si uvědomit, že dva stejné dokumenty mají stejný otisk. A to o otisk vytvořený algoritmem HMAC.

1.1.2 Proč HMAC není elektronický podpis?

HMAC (ani CMAC) nejsou algoritmy pro vytváření elektronického podpisu – nevytváří jedinečný důkaz, že HMAC (resp. CMAC) vytvořil Alois.

Vysvětlení je prosté. Kdyby Bohumila byla potvora, tak by zprávu od Aloise sama změnila a spočetla z ní HMAC. A Aloisovi by se vysmála. Kdyby se Alois divil, tak by mu ukázala změněnou zprávu a řekla mu: „Vidíš, co jsi zač, vždyť mne nemáš rád“. A Alois by se spravedlnosti nedovolal, protože by nemohl dokázat zdali HMAC opravdu spočetl on nebo jej podvrhla Bohumila.

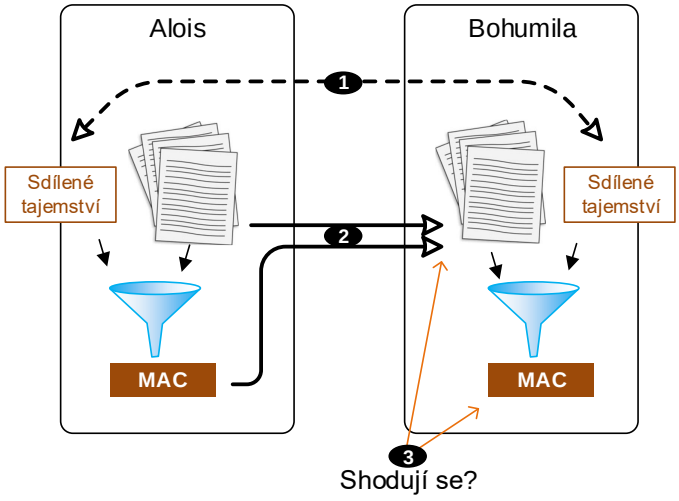
1.1.3 Replay attack, nonce

Uvedený mechanismus má jeden velký nedostatek. Útočník může odposlechnout a zaznamenat přenášená data zaslaná Aloisem Bohumile včetně HMAC. A po chvíli to celé Bohumile zaslat znovu (zopakovat). Tento typ útoku se označuje jako *replay attack*.

Pokud Alois zasílá Bohumile milostný dopis, pak útočník Bohumilu nanejvýš potěší. Avšak pokud Alois neposílá milostný dopis Bohumile, ale posílá platební příkaz, pak bude jeho platební příkaz zaplacen dvakrát a Alois přijde o peníze (v bankovnictví se to označuje jako *dual spend attack*).

Tomuto útoku se Alois brání např. pomocí vzestupného číslování svých zpráv. Pokud Bohumila obdrží zprávu nižšího než očekávaného čísla, pak pochopí, že se jedná o zopakovanou starou zprávu. Obdobně banka nezpracuje dva příkazy stejného čísla.

Jinou obranou je nonce. Nonce je dostatečně dlouhé náhodné číslo (zpravidla více jak 16 B dlouhé), které Alois vždy přidává do své zprávy (k přenášeným datům). Tím



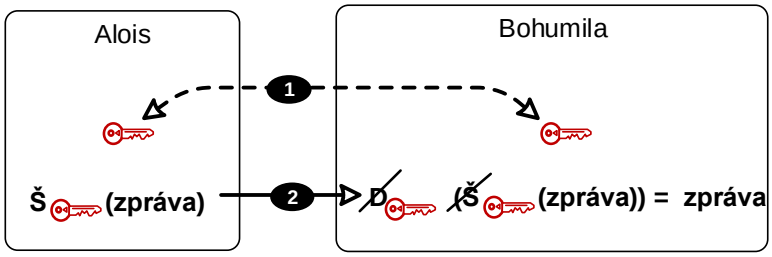
Obrázek 1.4 Zajištění integrity přenášených dat pomocí sdíleného tajemství

zajistí, že je nepravděpodobné, že by Alois odeslal dvě stejné zprávy a tudíž generoval dva stejné MAC. Avšak Bohumila musí kontrolovat, zdali již v minulosti neobdržela touž zprávou se stejnou nonce.

Jakou chybu může Alois udělat? Např. může použít chybný software, který negeneruje čísla náhodně. Pak

může vytvořit dvě stejné nonce. Aby si i tomuto předešlo, tak se někdy nonce vytváří tak, že se skládá ze dvou částí: jedna část obsahuje náhodné číslo a druhá část obsahuje datum a čas, který je sám o sobě neopakovatelný.

1.2 Symetrické šifry



Obrázek 1.6 Symetrická šifra

Jenže Alois si s Bohumilou mohou také přát, aby byl jejich vztahu zachován v tajnosti (aby byla zachována privátnost jejich vztahu), proto svou komunikaci šifrují. K dispozici mají např. symetrické šifry (Obrázek 1.6). Při výběru šifry si Alois s Bohumilou předem musí, na své schůzce, vyměnit tajný šifrovací klíč (Obrázek 1.6 šipka 1). Ten budou sdílet podobně, jako sdíleli sdílené tajemství. Nesmí dopustit, aby se k němu dostala třetí osoba (např. Cyril).

Alois zprávu šifruje tajným klíčem. Na výsledek pak Bohumila aplikuje dešifrovací algoritmus, pro který použije týž tajný klíč. Dešifrování se vyruší s šifrováním a Bohumila získá původní zprávu.

Symetrická šifra má v jistém smyslu autorizační účinek. Z pohledu Bohumily mohl zprávu zašifrovat pouze Alois, protože přece nikdo jiný než ona Alois nemají k dispozici tajný klíč.

Symetrické šifry se dělí na blokové a proudové.

1.3 Blokové šifry

Blokové šifry data se šifrují/dešifrují po blocích pevné délky. Pokud vstupující data jsou kratší, pak se data musí dorovnat na délku bloku.

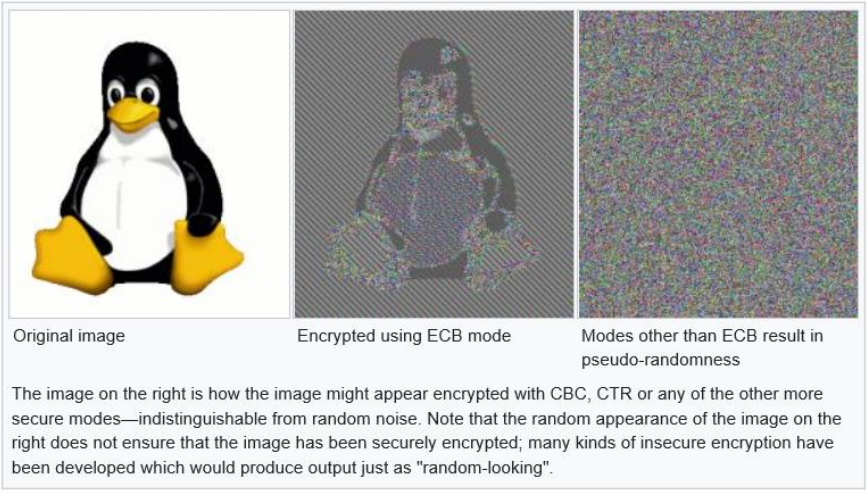
1.3.1 Mód blokové šifry

Jenže šifrovaná zpráva se může skládat z více bloku. Způsob, jakým se vzájemně váže šifrování jednotlivých bloků se nazývá modem šifry. Nejjednodušším módem je ECB (*Electronic Codebook*), kdy se jednotlivé bloky šifrují samostatně (nezávisle na ostatních blocích). Tj. ECB šifrované bloky nikterak vzájemně neváže.

I když útočník nevidí do šifrovaného textu, tak v případě módu ECB může útočit tak, že zamění pořadí jednotlivých zašifrovaných bloků. Pak může místo částky 0000.0001 Kč zaplatit 1000.000 Kč. A to je rozdíl. Rovněž zajímavý je i Obrázek 1.5, který ukazuje příklad šifrování módem ECB obrázku ve formátu bitové mapy, které obsahuje velké stejnobarevné oblasti.

Velice zajímavou otázkou je, jakou informaci máme zahrnout do prvního šifrovaného bloku. Pokud bychom nezahrnovali nic, pak by dva shodou okolností stejné texty měly i shodné zašifrované texty. Útočník by sice nebyl schopen získat původní (nešifrovaný) text, ale informace, že se jedná o touž zprávu, pro něj také nemusí být k zahození. Proto se často před šifrováním zprávy vygenerují náhodná data (tzv. inicializační vektory), které se zahrnou do šifrování prvního šifrovaného bloku.

Na inicializační vektory jsou kladeny jiné bezpečnostní požadavky než na tajné šifrovací klíče. Inicializační vektory se totiž nemusí utajovat.

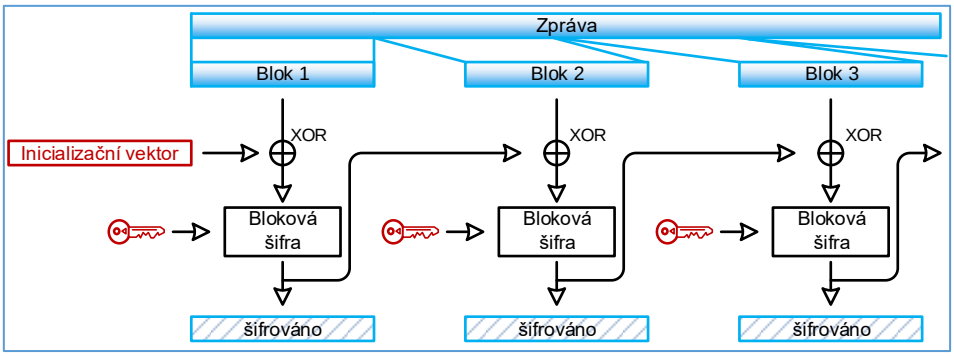


Obrázek 1.5 Šifrování bitové mapy (převzato z Wikipedie) v ECB módu

Šifra	Délka klíče	Délka bloku
DES	56 + 8 paritních bitů	64
IDEA	128	64
RC5	až 2040	32, 64 nebo 128
Blowfish	1-448	64
Kasumi	128	64
Twofish	až 256	128
Camellia	128	128
Serpent	128, 192 nebo 256	128
AES	128, 192 nebo 256	128

Tabulka 1.2 Některé blokové šifry

Velice známým módem blokových šifer je mód CBC (Obrázek 1.7). V tomto případě se nejprve na první blok operací XOR aplikuje inicializační vektor. Na další blok, se před jeho šifrováním operací XOR, aplikuje výsledek šifrování předchozího bloku atd. Mód CBC byl publikován



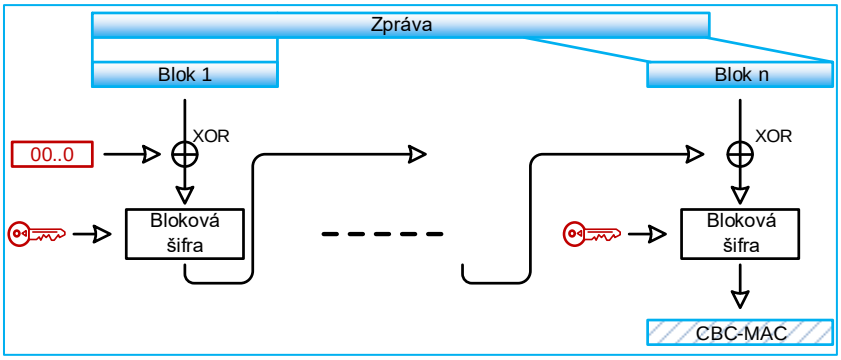
Obrázek 1.7 Bloková šifra v CBC módu

v roce 1976 a dnes již není příliš doporučován, protože na něj byly publikovány útoky.

1.3.2 CMAC

Pakliže si prohlédneme Obrázek 1.7 tak si uvědomíme, že poslední blok je ve své podstatě otiskem celé šifrované zprávy. Navíc závislý na hodnotě tajného klíče.

Na obrázku Obrázek 1.8 je pak znázorněn princip algoritmu CBC-MAC, který je též algoritmem na ochranu integrity – obdobně jako HMAC. Tento algoritmus používá jako inicializační vektor binární nuly. (Obrázek neřeší problém doplnění posledního bloku v případě, že délka zprávy není dělitelná délkou šifrovacího bloku.).



Obrázek 1.8 CBC-MAC

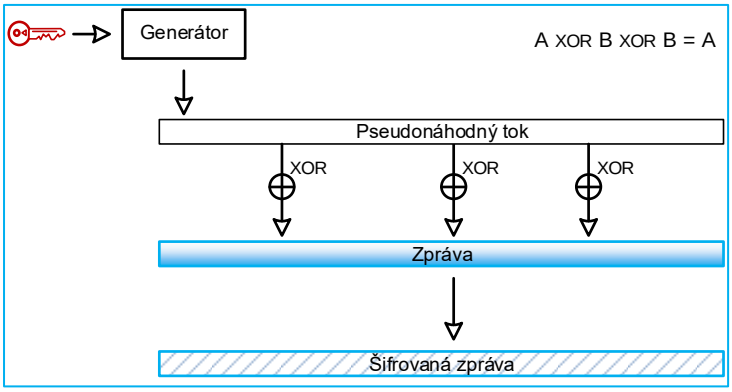
NIST jako jeden z algoritmů pro ochranu integrity doporučil algoritmus CMAC, který je variací algoritmu CBC-MAC.

1.3.3 Proudové šifry

Myšlenka proudové šifry vychází z vlastnosti logické operace XOR:

A XOR B XOR B = A

Základem je generátor pseudonáhodného toku dat, jehož výstup je deterministicky závislý na hodnotě tajného klíče.



Obrázek 1.9 Proudová šifra

Výstup z generátoru (to je to A) se pak operací XOR bit po bitu aplikuje a zprávu (to je B), kterou chceme šifrovat. Výsledkem je zašifrovaná zpráva.

V případě dešifrování se na zašifrovanou zprávu (tj. na A XOR B) aplikuje pseudonáhodný tok (B). Výsledkem je dešifrovaná zpráva (A). Nutné je použít týž generátor se stejným tajným klíčem.

Trochu jiný přístup využívá CTR (*Counter*) mód blokových šifer (Obrázek 1.10). V tomto případě se za využití blokové šifry vytváří pseudonáhodný tok dat.

Šifrovaná zpráva se opět rozdělí na bloky. Na každý blok se operací XOR aplikuje blokovou šifrou šifrovaná dvojice Nonce + čítač. Tato dvojce má délku šifrovacího bloku.

Nonce je jedinečný řetězec pro dané šifrování a čítač se zpravidla po jedné zvyšuje tak, jak jednotlivé bloky vstupují do šifrování.

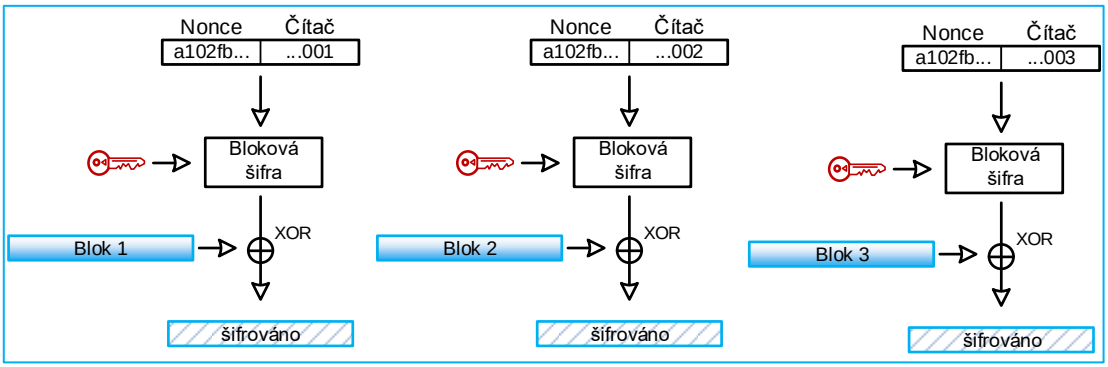
Algoritmus AES s modem CTR se využívá např. v případě šifrování multimediální komunikace mezi mobilním zařízením a základnovou stanicí. V telefonním hovoru se totiž může sem tam nějaký paket ztratit, což algoritmu AES-CTR nevadí.

Tabulka 1.3 Některé proudové šifry

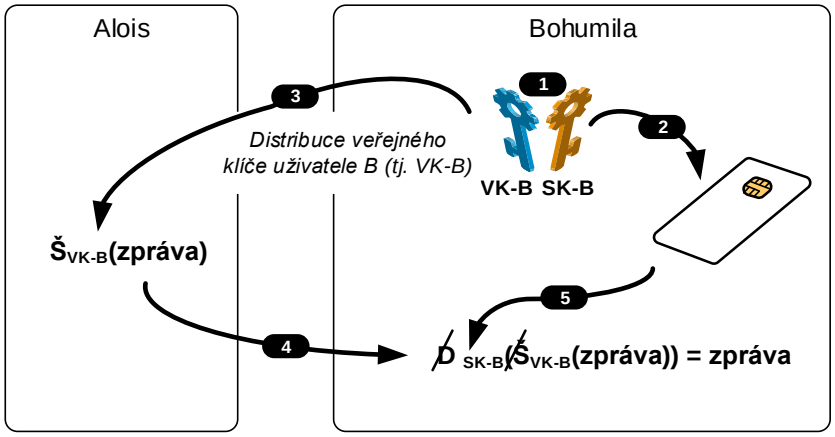
Šifra
SNOW 2.0
SNOW 3G
AES-CTR
ZUC

1.4 Asymetrické šifry

Jiným typem šifer jsou asymetrické šifry. Tyto šifry nepoužívají jeden tajný šifrovací klíč sdílený mezi odesilatelem a příjemcem, ale vždy se používá pár šifrovacích klíčů. Jeden klíč pro šifrování a druhý pro dešifrování. U elektronického podpisu pak uvedeme, že operace



Obrázek 1.10 Bloková šifra v CTR módu



Obrázek 1.12 Asymetrická šifra

šifrování a dešifrování jsou u některých šifer zaměnitelné, proto u asymetrických šifer nemluvíme o šifrovacím a dešifrovacím klíči, ale o veřejném a soukromém klíči. Asi nejznámějším asymetrickým šifrovacím algoritmem je algoritmus RSA.

Pokud chce Alois šifrovat zprávu Bohumile asymetrickou šifrou pak (Obrázek 1.12):

1. Bohumila, tj. příjemce zprávy, si musí vygenerovat dvojici klíčů: veřejný klíč (VK-B) a soukromý klíč (SK-B).
2. Bohumila si uloží svůj soukromý klíč do důvěryhodného úložiště klíčů. Např. na disk, na čipovou kartu atd. Soukromý klíč je aktivem Bohumily, které si musí střežit.
3. Bohumila distribuuje svůj veřejný klíč VK-B do celého světa. Klidně může svůj veřejný klíč poslat Aloisovi po žárlivém Cyrilovi.
4. Alois po obdržení veřejného klíče Bohumily šifruje zprávu Bohumile jejím veřejným klíčem (VK-B).
5. Bohumila (příjemce) dešifruje přijatou šifrovanou zprávu svým soukromým klíčem SK-B a získá původní zprávu.

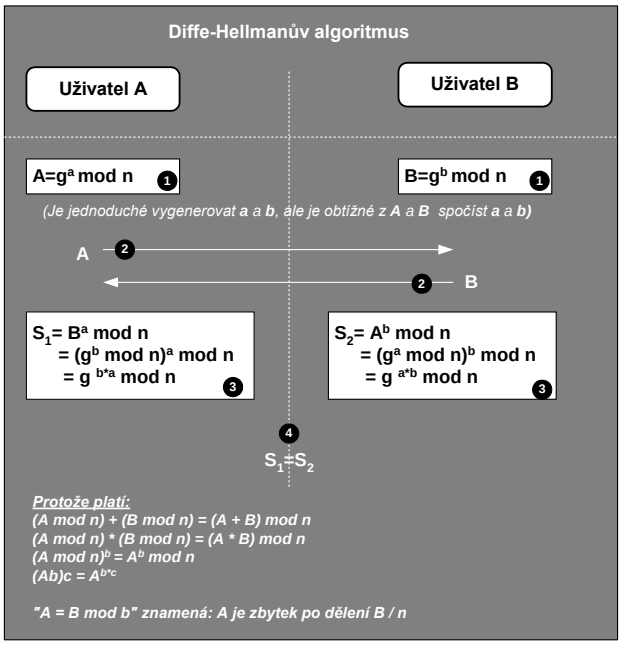
Základní vlastností šifrování na bázi asymetrických algoritmů je skutečnost, že je relativně jednoduché za využití veřejného klíče šifrovat text, ale na základě znalosti tohoto veřejného klíče a veřejným klíčem šifrované zprávy je velice obtížné získat původní zprávu.

Délka šifrovacích klíčů pro algoritmus RSA se tč. považuje za ještě bezpečnou, pokud je alespoň 2048 bitů.

Existují i jiné asymetrické algoritmy. Dnes se často mluví o algoritmu ECC (eliptické křivky). Obecně se míní, že z bezpečnostního hlediska odpovídá 2048 bitů dlouhému RSA klíči 224-256 bitů dlouhý ECC klíč (podle typu ECC algoritmu).

1.5 Algoritmy pro výměnu klíčů

Jiným algoritmem je Diffie-Hellmanův (DH) algoritmus. Ten se vůbec nehodí k nějakému asymetrickému šifrování, ale k bezpečnému ustavení tajných klíčů či sdílených tajemství. Aby Alois s Bohumilou mohli komunikovat symetrickou šifrou, tak se nejprve za využití algoritmu DH dohodnou na tajném klíči, aniž by museli organizovat nějakou tajnou schůzku. Oba nejprve vygenerují dvojici: veřejné a soukromé DH číslo. Vzájemně si pak vymění (např. volně přes Internet) svá



Obrázek 1.11 Diffie-Helmanův algoritmus

veřejná DH čísla. Obě strany jsou následně ze znalosti svého veřejného a soukromého DH čísla a veřejného DH čísla svého protějšku schopny spočíst sdílené tajemství. Od tohoto tajemství je pak snadno možné nějakou transformací odvodit symetrický šifrovací klíč, který se použije pro šifrování vzájemné komunikace např. algoritmem AES. Diffie-Hellmanův algoritmus hojně využívá např. IPsec.

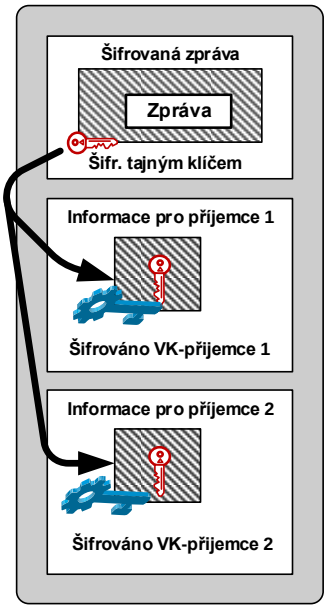
1.6 Elektronická obálka

Bez ohledu na délku klíčů obecně platí, že asymetrické šifrovací algoritmy jsou výpočetně mnohem náročnější, než symetrické algoritmy.

Šifrování je vždy operací, do které vstupují data určená k zašifrování a šifrovací klíče (a někdy též inicializační vektory). V případě využití asymetrické kryptografie, kde se používají výpočetně náročné matematické postupy, je doba trvání výpočtu dlouhá.

Např. klíč RSA o délce jen 1024 bitů se v desítkové soustavě zapíše jako číslo s více než 300 ciframi a nelze tak použít standardních operací mikroprocesoru.

Řešením tohoto problému je elektronická obálka (Obrázek 1.13). Odesilatel zašifruje zprávu náhodným tajným (symetrickým) klíčem, což je rychlá operace. A k takto šifrované zprávě jen pro každého příjemce přibalí „informace pro příjemce“ (*Recipient info*) obsahující mj. náhodný tajný klíč zašifrovaný veřejným klíčem příjemce. Takže asymetricky se šifruje pouze kraťoučký tajný klíč. Výsledek je velice rychlý a efektivní. Má to ještě jeden pozitivní efekt. Pokud zprávu posíláme více adresátům, pak ji šifrujeme pouze jednou náhodným tajným klíčem a každému adresátovi ke zprávě přibalíme tajný klíč šifrovaný jeho veřejným klíčem. Tj. pokud náš Alois má kromě Bohumily ještě další milenky, pak může vyznání lásky rozeslat jen jednou, přičemž pro každou z milenek vyznání šifruje tajným veřejným klíčem odpovídající milenky. Toho využívá zabezpečení elektronické pošty protokolem S/MIME.

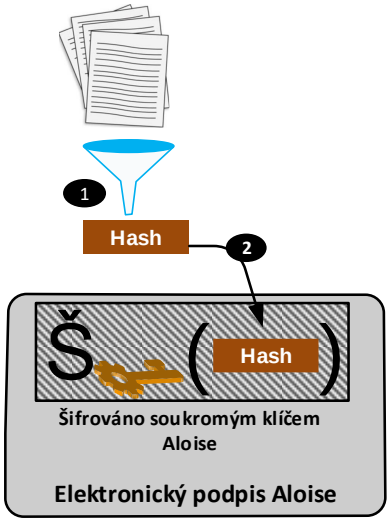


Obrázek 1.13 Elektronická obálka

1.7 Elektronický podpis

Elektronický (nebo též elektronický) podpis je mechanismus, kterým se zajišťuje důkaz nepopiratelnosti dat (pravosti dokumentů). Elektronický podpis (Obrázek 1.14) se vytváří v následujících krocích (Obrázek 1.15):

1. Alois spočte otisk z dokumentu.
2. Výsledný otisk šifruje svým soukromým klíčem. Soukromým klíčem šifrovaný otisk zprávy se nazývá elektronickým podpisem zprávy.
3. Dvojici dokument a jeho elektronický podpis Alois odešle Bohumile.



Obrázek 1.14 Elektronický podpis

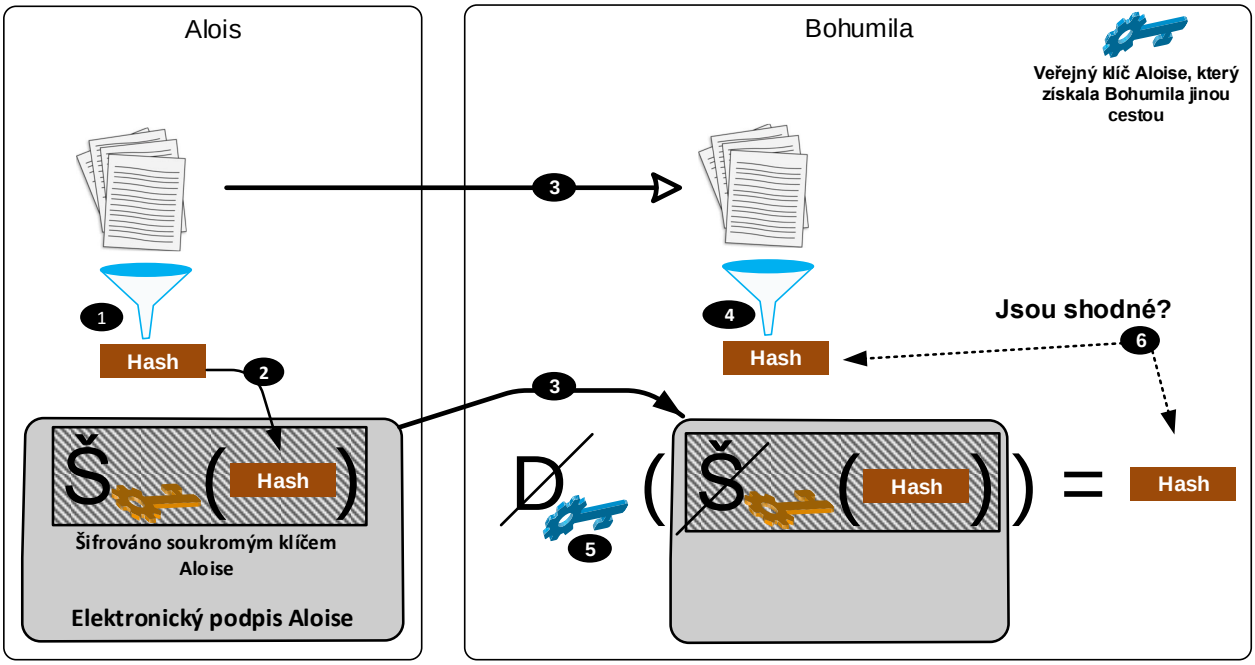
Bohumila pak verifikuje elektronický podpis:

4. Bohumila samostatně spočte otisk z přijaté zprávy.
5. Bohumila dešifruje přijatý elektronický podpis veřejným klíčem Aloise.
6. Bohumila porovná výsledek získaný z bodu 4 s výsledkem získaným z bodu 5. Pokud jsou stejné, pak elektronický podpis mohl vytvořit pouze ten, kdo vlastní soukromý klíč Aloise – tedy Alois a navíc tato skutečnost prokazuje, že zpráva nebyla během přenosu pozměněna, tj. zajišťuje integritu zprávy.

Elektronický podpis provádí důkaz pravosti na základě vlastnictví soukromého klíče.

Je tedy nutné, abychom si své soukromé klíče dobře střežili. Ztráta soukromého klíče je pak obdobná výměně podobizny v občanském průkazu; či záměně otisků prstů v evidenci pachatelů. Neopatrnost ochrany soukromého klíče lze přirovnat k podepsání bianco šeků.

Na rozdíl od šifrování, elektronický podpis použije klíč toho, kdo zprávu podepisuje (nikoliv příjemce jako u šifrování). Mlčky jsme tedy předpokládali, že náš algoritmus umožňuje nejprve „dešifrovat“ soukromým klíčem a pak



Obrázek 1.15 Verifikace elektronického podpisu

„šifrovat“ veřejným klíčem. Tj. že operace šifrování a dešifrování jsou zaměnitelné. Takovým algoritmem, který takovouto záměnu umožňuje, je např. algoritmus RSA.

1.8 Prokazování totožnosti (autentizace) na základě asymetrické kryptografie

Cílem prokázání totožnosti je důkaz, že daná osoba vlastní příslušný soukromý klíč. Takový důkaz lze vytvořit na základě šifrování i elektronického podpisu. V případě šifrování se osobě zašle jejím veřejným klíčem šifrovaný náhodný řetězec. Když osoba vlastní příslušný soukromý klíč, tak tento řetězec umí dešifrovat.

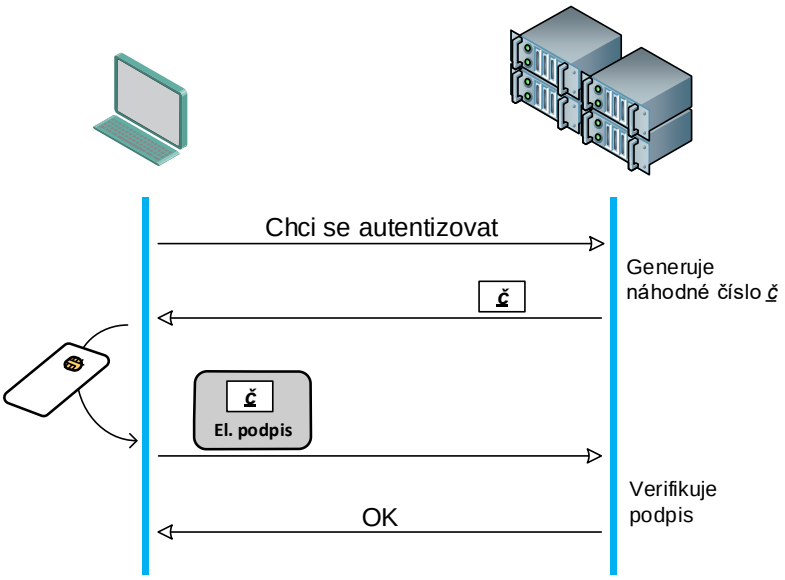
V případě elektronického podpisu. Uživatel opět prokazuje svou totožnost tím, že dokáže, že vlastní příslušný soukromý klíč a je schopen jej použít. Obrázek 1.16 takovou autentizaci schématický vyjadřuje:

- Uživatel se chce autentizovat vůči serveru, tak mu pošle svou identitu.
- Server generuje náhodný řetězec č a zašle jej klientu.
- Klient náhodný řetězec elektronicky podepíše a vrátí serveru.
- Server verifikuje podpis (podle deklarované identity klienta najde v databázi klientů příslušný veřejný klíč). V případě, že verifikace proběhne s kladným výsledkem, tak je klient autentizován, jinak je autentizace zamítnuta.

Obě zmíněné metody autentizace mají řadu neduhů. Ve své podstatě se jedná o autentizaci jednorázovým heslem. Jednorázové heslo je vždy příležitostí pro muže uprostřed, který se ho snaží získat dříve, než dorazí na server, aby jej využil ve vlastní prospěch (Obrázek 1.19).

Důležité také je, aby se podepisovaný/šifrovaný text nemohl opakovat (byl např. náhodný). Pokud by se připustilo k opakování, pak opět byl možný *reply attack*.

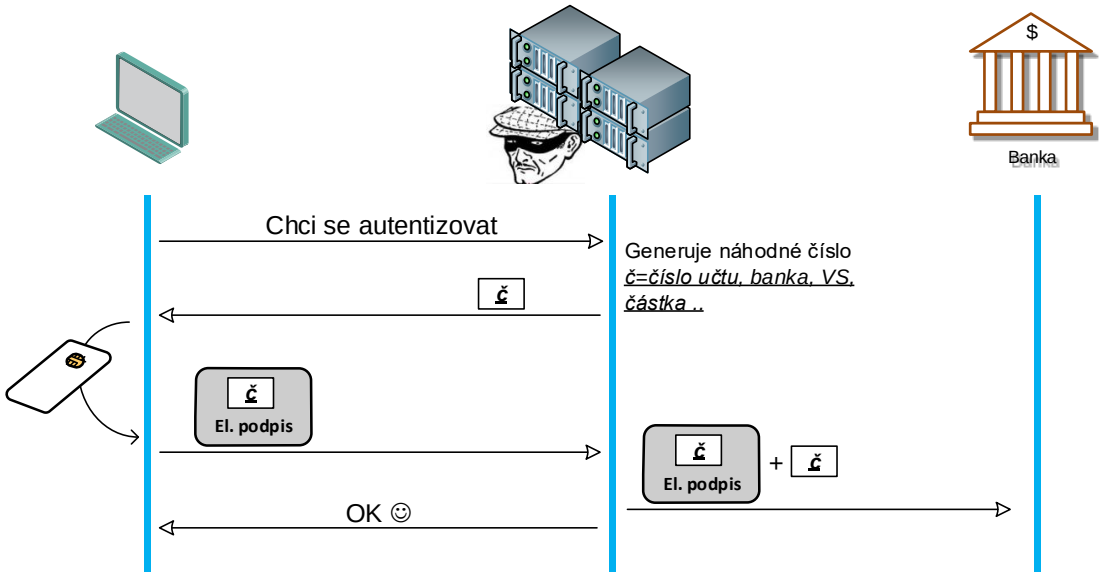
Použití elektronického podpisu k autentizaci má ještě jeden zajímavý neduh. Podepisovaný řetězec musí být značně velký. Tj. může se do něj např. vejít platební příkaz (Obrázek 1.17). Podvržený server pak může vyžadovat autentizaci, kdy místo náhodného řetězce použije platební příkaz, který následně zneužije.



Obrázek 1.16 Autentizace elektronickým podpisem

Poučení z tohoto plyne, že Elektronický podpis jako algoritmus může být využíván jednak pro autentizaci uživatele (zpravidla podepisuje náhodný řetězec) a jednak pro elektronický podpis dokumentů jako důkaz pravosti dokumentu. Obrázek 1.17 jasně ukazuje, že z bezpečnostního pohledu je rizikem používat týž pár veřejný/soukromý klič pro podpis i pro šifrování.

Protokoly SSL/TLS využívají autentizaci klientů na bázi asymetrické kryptografie, ale nikoliv takto triviálně. Důsledně dbají na to, aby se ověřilo, že autentizovaná strana je opravdu vlastníkem soukromého klíče.



Obrázek 1.17 Útok na autentizaci pomocí elektronického podpisu

Pro občana se tak základem stává ochrana jeho soukromého klíče, neboť soukromý klíč je jeho cenným aktivem. Odcizení soukromého klíče by způsobilo to, že zloděj by se mohl elektronicky prokazovat místo majitele soukromého klíče. Opět připomeňme, že odcizení soukromého klíče lze přirovnat v případě klasických občanských průkazů k odcizení podoby z fotografie občanského průkazu.

1.9 Tři typy asymetrických klíčů

Elektronický podpis jako algoritmus může být využíván jednak pro autentizaci uživatele a jednak pro elektronický podpis dokumentů jako důkaz pravosti dokumentu.

V předchozím paragrafu jsme obhajovali nutnost samostatných párů soukromý/veřejný klíč pro:

- Elektronický podpis dokumentů
- Autentizaci uživatele

Nicméně potřebuje ještě další pár soukromý/veřejný klíč:

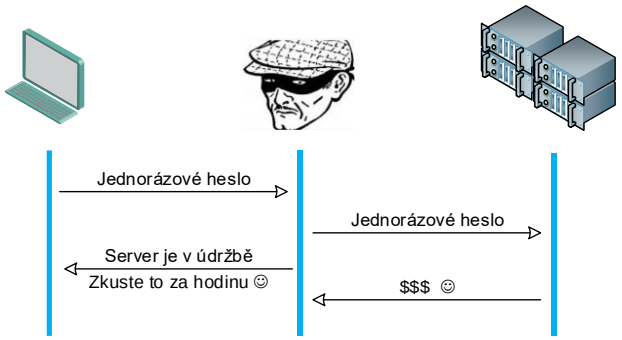
- Pro šifrování dokumentů (přesněji pro elektronickou obálku)

Proč potřebujeme třetí pár pro šifrování? Přinejmenším máme dva pádné důvody:

- První důvod je kryptografický. Dobrou zprávou pro hackera lámajícího šifru totiž je, že má k dispozici známý text šifrovaný soukromým klíčem (elektronický podpis) a jiný známý text šifrovaný veřejným klíčem (např. náhodný symetrický klíč v elektronické obálce).
- Druhým, ale pádnějším, důvodem je praktické používání soukromého klíče. Jestliže uživatel ztratí soukromý klíč (nikoliv vyzradí) určený k elektronickému podpisu nebo k autentizaci, pak se vcelku nic neděje. Soukromý klíč je totiž třeba pouze pro vytváření nového podpisu (existující podpisy se verifikují pomocí veřejného klíče). Uživatel si v takovém případě vygeneruje nový pár klíčů a může podepisovat další dokumenty. Soukromý klíč určený k vytváření podpisu netřeba archivovat, ale pouze uchovávat na nosičích, které soukromý klíč nikdy nemůže opustit (např. na čipové kartě). Nikdy takový klíč nedáváme z ruky, protože by jím mohly být podepisovány dokumenty v náš neprospěch.

Obrázek 1.18 Autentizační karta

Kdežto šifrovací klíč je mnohdy dobré zálohovat, abychom jej měli i v případě ztráty primárního úložiště sou-



Obrázek 1.19 Vylákaní hesla

kromého klíče. V případě ztráty soukromého klíče určeného pro šifrování (přesněji pro dešifrování) totiž ztratíme veškeré tímto klíčem zabezpečené dokumenty, protože je nemáme čím dešifrovat.

Jelikož mají páry pro šifrování a elektronický podpis různé životní cykly, tak je praktické mít samostatné páry šifrovacích/dešifrovacích klíčů pro šifrování, podepisování i pro autentizaci.

1.10 Autentizační metody založené na jiných principech

Mnohdy se autentizace (prokazování totožnosti) spojuje s přihlášením uživatele k počítačovému systému. Klasická autentizace v počítačovém světě byla orientována na autentizaci pomocí jména a stálého hesla, která jsou lákadlem pro muže uprostřed, který je může jednak odposlechnout a jednak vylákat.

V dnešní době je problém autentizace uživatele aktuální pro internetové a mobilní aplikace, které používá široká veřejnost. A tak pro volbu autentizace začínají platit i kritéria, která byla dříve spíše v pozadí. Jedná se např. o cenu autentizačních pomůcek či pracnost autentizace pro klienta. Stačí se nad problémem zamyslet prakticky: je rozdíl v tom, nakupuje-li firma autentizační kalkulátory v ceně několika set Kč pro padesát zaměstnanců nebo pro padesát tisíc klientů.

Na pracnost autentizace je možno se dívat ze dvou pohledů: z pohledu pracnosti a z pohledu nároků na znalosti nutné k provádění autentizace. Mechanická pracnost vstupuje do hry např. při použití autentizačních kalkulátorů, kdy klient musí do kalkulátoru a z kalkulátoru přepisovat data. Při použití elektronického podpisu je autentizace pro uživatele mechanicky jednoduchá, avšak uživatel musí pochopit princip elektronického podpisu, musí obnovovat certifikáty atd.

Autentizaci uživatele je možné provést na základě prokázání:

- **že uživatel něco má** (autentizační kalkulátor, čipovou kartu či v poslední době mobilní telefon);
- **že uživatel něco ví** (heslo, PIN);
- **že uživatel něčím je** - má např. nějaké biometrické vlastnosti (otisky prstů, struktury oční sítnice či duhovky, tvar obličeje, atp.).
- **že uživatel něco umí** (podepsat se)

Snahou je pak jednotlivé uvedené metody kombinovat.

Vedle autentizace (prokazování totožnosti) budeme používat ještě termín autorizace. Zatímco autentizací prokážeme, o koho se jedná, tak autorizací budeme konkrétnímu subjektu přiřazovat role a z nich plynoucí oprávnění, která má v jednotlivých aplikacích. Např. uživateli Fr. Novákovi poté co prokáže svou totožnost (se autentizuje), tak je mu v aplikaci XY poskytnuta role administrátora. Tj. F. Novák je pro aplikaci XY autorizován jako administrátor.

V PKI se pro autentizaci využije certifikát veřejného klíče a pro autorizaci pak atributové certifikáty, které se příliš nerozšířily. Tato kapitola je však věnována jiným autentizačním metodám než certifikátům, aby čtenář získal pokud možno objektivní porovnání jednotlivých autentizačních metod.

1.10.1 Stálá hesla

Přístupové heslo je typickým příkladem stálého hesla. Na straně serveru nebývá uchováváno v čisté textové podobě, ale znehodnocené jednocestnou funkcí proti zneužití správcem systému.

V okamžiku, kdy uživatel zadá heslo, aby se autentizoval, systém zavolá na zadané heslo jednocestnou funkci a výsledek se porovná s údajem uloženým v systému. Algoritmů jednocestných funkcí je celá řada. Nejčastěji jsou založeny buď na výpočtu otisku, nebo na symetrické šifře.

Stále heslo může být:

- Odposlechnuto zejména v případě, že je přenášeno po nezabezpečených spojích.
- Vylákáno pomocí podvrženého serveru (Obrázek 1.19).

1.10.2 Jednorázová hesla

Jednorázová hesla řeší problém odposlechu hesla během jeho přenosu sítí a následným použitím odposlechnutého hesla. Jednorázové heslo neřeší problém vylákání (Obrázek 1.19).

Nesmíme, ale zapomínat např. v pomocí SSL/TLS zabezpečených relacích lze vylákat heslo v případě, že je využita autentizace serveru, ale není využita autentizace klienta certifikátem.

Pokud je jednorázové heslo využito pouze pro počáteční autentizaci, pak ještě po úspěšně proběhlé autentizaci existuje nebezpečí v převzetí relace útočníkem. Proto se jednorázová hesla zpravidla ještě následně využívají pro další zabezpečení relace (např. pomocí doplňování MAC

k blokům přenášených dat či jako součást symetrických klíčů pro šifrování relace).

Jednorázová hesla se nepoužívají pouze u aplikací provozovaných v počítačových sítích, ale i u tak odlišných aplikací, jako je CallCentrum, kdy je nutné autentizovat uživatele, který požaduje služby běžným telefonem.

Jak je vlastně možné, že uživatel může pokaždé zadat jiné heslo? Algoritmů na tvorbu jednorázových hesel je celá řada.

Seznam jednorázových hesel

Nejjednodušší metodou jednorázových hesel je seznam jednorázových hesel. V tomto případě je vygenerován seznam hesel, který může být vytištěn na papír a předán uživateli (resp. zaslán uživateli bezpečnou elektronickou poštou). Stejný seznam existuje i na straně systému, kde mohou být i jednotlivá jednorázová hesla znehodnocena jednocestnou funkcí.

Uživatel pak pro svou autentizaci zadává jedno heslo po druhém. Po zadání hesla si jej škrtne ze seznamu.

Jednorázová hesla mohou být v seznamu i očíslována. Systém pak může ve výzvě pro zadání hesla napovědět uživateli, jaké heslo má zadat.

Jednou z nevýhod tohoto způsobu je, že po vyčerpání seznamu musí být uživateli vygenerován a předán další seznam. Další nevýhodou seznamu hesel je, že si jej uživatel těžko může zapamatovat, a tak jej musí nosit s sebou v tištěné či elektronické podobě. Může se tak snadno stát, že uživatel seznam jednorázových hesel někde zapomene.

Seznamy jednorázových hesel se často kombinují s klasickým heslem. Uživatel pak zadává heslo skládající se ze dvou částí: ze stálého hesla („PINu“) a z jednorázového hesla. Tím se komplikuje využití seznamu jednorázových hesel zapomenutého v internetové kavárně a na druhou stranu se i komplikuje použití odposlechnutého hesla.

Jinou možností používání jednorázových hesel je jednotlivá hesla očíslovat a nevyžadovat hesla jedno po druhém, ale náhodně. Náhodný výběr hesel může být i s opakováním, tj. pak se v podstatě nejedná o „jednorázové heslo“, ale požadavek na výběr dvou stejných hesel v pro útočníka zajímavém čase je málo pravděpodobný.

Pro některé aplikace je dostatečná i autentizační karta (Obrázek 1.18). Jedná se o plastikovou kartičku bez magnetického proužku a bez čipu s několika předtištěnými sadami čísel.

Princip použití spočívá v tom, že aplikace vygeneruje dotaz na zadání několika náhodně vybraných čísel vytištěných na kartě. Např. v podobě „zadejte třetí čtveřici čísel ze čtvrté sady vytištěných na vaší Autentizační kartě“.

Autentizační karta je forma použití vícenásobných hesel s jednoduchým doplňkem vzdáleně připomínajícím heslo na jedno použití.

Výhodou tohoto prostředku jsou jeho zanedbatelné pořizovací náklady, kterými jsou získány velmi zajímavé bezpečnostní vlastnosti.

Rekurentní algoritmus

Rekurentní algoritmus využívá jednocestné funkce (např. otisk). Zvolme si konkrétní jednocestnou funkci, kterou označíme jako F. Dále si uživatel musí sám zvolit nějaký počáteční řetězec *násada*. Tento řetězec uživatel nikomu nesděluje – je to uživatelovo tajemství.

Jednocestnou funkci F aplikovanou na řetězce *násada* vyjádříme jako:

F(*násada*).

Použijeme-li algoritmus F dvakrát opakovaně na tutéž zprávu, tj. F(F(*násada*)), pak budeme psát:

F_2(*násada*)

a obdobně

F_n(*násada*)

bude znamenat, že jsme použili algoritmus F na řetězec *násada* celkem n-krát.

Použití této metody spočívá též nejprve v inicializačním kroku, kdy:

- Uživatel vygeneruje text *násada*.
- Uživatel a správce aplikace se dohodnou na čísle *n*, např. 1000. Uživatel vyrobí: F_{1000}(*násada*) a předá jej správci aplikace. Správce aplikace si do databáze k našemu uživateli poznamená název algoritmu jednocestné funkce (tj. F), číslo 1000 a hodnotu F_{1000}(*násada*). Správce však nezná hodnotu řetězce *násada* (je to uživatelovo tajemství).

Při autentizaci pošle uživatel na server jméno, server ve své databázi zjistí, jakou uživatel používá autentizační metodu (F). Obratem server uživateli pošle dotaz obsahující číslo *(n-1)*, tj. nyní 999. Uživatel vygeneruje odpověď F_{999}(*násada*) a odešle je jako jednorázové heslo serveru. Server prověří totožnost uživatele, tím, že provede porovnání:

$F(F_{999}$(*násada*)) = F_{1000}(*násada*)

Algoritmus F je mu znám a hodnotu F_{1000}(*násada*) má uloženu v konfiguračním souboru a F_{999}(*násada*) obdržel v odpovědi uživatele.

Po úspěšné autentizaci uživatele uloží server do databáze místo hodnoty F_{1000}(*násada*) hodnotu F_{999}(*násada*) a

místo čísla 1000 číslo 999. Při další autentizaci se vše provádí s číslem o jedničku nižším, tj. provádí se autentizace:

$F(F_{998}(\textit{násada})) = F_{999}(\textit{násada})$

Uživatel mohl tedy vygenerovat celkem 999 hesel na jedno použití, pak musí změnit hodnotu řetězce *násada* a správci serveru předat nový $F_{1000}(\textit{násada})$.

Sdílené tajemství

Obrázek 1.4 znázorňuje i princip autentizace za využití sdíleného tajemství. Obrázek ale znázorňuje důkaz pravosti dokumentu. V tomto případě je využit HMAC (mohl by být využit i CMAC). Pokud chceme HMAC využít nikoliv pro autentizaci dokumentu ale pro autentizaci osoby, pak základním problémem je s jakými daty řetězit sdílené tajemství, tj. z čeho počítat otisk.

Cílem tedy je generovat jednorázová hesla, jež budou spočtena jako HMAC. Vlastně jsme v absurdní situaci. Víme, jaký má být výsledek, víme, že jej budeme počítat z něčeho, co budeme řetězit se sdíleným tajemstvím, ale nevíme z čeho. Navíc by bylo nepříjemné, kdyby bylo pravděpodobné, že generovaná hesla se budou opakovat.

Musíme tedy najít něco, co zná jak autentizovaný (klient), tak i server, který bude autentizaci verifikovat. Takovými veličinami jsou např.:

- Datum a čas. Stačí si představit elektronické hodinky s minutovou přesností. Pokud se na nich zobrazuje datum a čas, pak se tato hodnota během dne neopakuje a platí nejvýše jednu minutu. V takovém případě se jako text zprávy pro výpočet HMAC může vzít datum a čas. Drobnou závadou je časová odchylka hodin uživatele a serveru.

  Tento algoritmus je principem mnohých autentizačních kalkulátorů. Časová odchylka kalkulátoru a serveru se kompenzuje tím, že server ještě vyzkouší jednorázová hesla generovaná z okolních časů, pakliže některé vyjde, pak uživatele nejenom autentizuje, ale příslušnou odchylku si uloží do databáze.

- Počet přihlášení uživatele k systému je rovněž stále monotónně vzrůstající posloupností. Drobnou potíží jsou stavy, kdy se klientovi přeruší komunikace během autentizace.
- Náhodné číslo generované serverem. V podstatě se jedná o symetrickou obdobu autentizace asymetrickým algoritmem (Obrázek 1.16). Jedná se o dialog dotaz-odpověď (*challenge-response*):
  - Server generuje náhodný řetězec a odešle jej klientovi jako dotaz.
  - Klient zřetězí dotaz se sídlením tajemstvím a na výsledek aplikuje HMAC. Výsledkem je jednorázové heslo, které jako odpověď klient vrátí serveru.

Symetrická šifra

Ke generování jednorázových hesel se může místo otisku využít též symetrickou šifru. Namísto sdíleného tajemství sdílí klient se serverem symetrický šifrovací klíč, který využije k šifrování: času, počtu přihlášení či náhodně generovaného dotazu.

Jednorázové heslo doručované přes nezávislý kanál

Principem této metody je, že dotaz generuje server, který jej uživateli doručí nezávislým kanálem. Druhým kanálem může být fax, e-mail, mobil apod. Každý z použitých kanálů přitom nemusí být příliš bezpečný, útočník by ale musel prolomit dva na sobě nezávislé komunikační kanály současně (v jednom okamžiku), což je velice těžko uskutečnitelné. Tento princip se také nazývá principem dvou zámků. Předpokládejme, že druhým kanálem je mobil.

Důležité je, aby nezávislý kanál byl opravdu nezavilý, proto s nástupem chytrých telefonů zasílání hesel skrze SMS ztrácí smysl.

Autentizační kalkulátory

Tento způsob autentizace kombinuje dva nezávislé komunikační kanály. Tato skutečnost podstatným způsobem omezuje možnost zneužití, protože případný útok by musel být veden společně na oba nezávislé kanály, což je vysoce náročné. Další podstatnou výhodou jsou nízké pořizovací náklady a relativně jednoduchá obsluha. Jistým omezením tohoto řešení je, že klient musí být vybaven autentizačním kalkulátorem.

Nevýhodou autentizačních kalkulátorů totiž je samotná existence kalkulátoru, tj. z hlediska provozovatele aplikace se kalkulátor musí pořídit, což není laciné. Z hlediska uživatele je zase nepříjemné, že se kalkulátor musí stále nosit s sebou, a přitom je dobré jej nezničit či neztratit.

Nahrazení autentizačních kalkulátorů mobily je obchodním snem. Existuje mnoho rizik, která je třeba zvážit před tím, než implementujeme autentizaci mobilním telefonem.

Biometrika

Biometrických vlastností člověka je možné měřit celou řadu. Ekonomicky nejpřijatelnější jsou zatím stále otisky prstů. Navíc data popisující otisk prstů lze omezit na 300 až 600 bajtů. Nevýhodou otisků prstů je skutečnost, že se jedná o biometrická (citlivá) osobní data, a musí tak s nimi být i zacházeno. Navíc se v komerční praxi používá řada formátů dat popisujících otisky prstů, a tak zařízení různých výrobců nemusí být vzájemně kompatibilní.

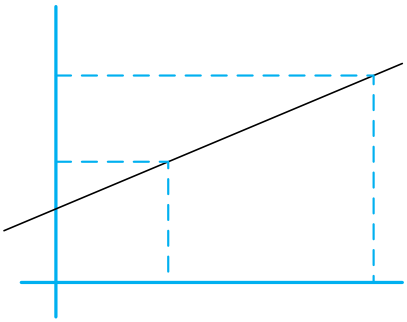
Náš pohled je však z pohledu sítě. A z tohoto pohledu není ani tak zajímavé, že je možné snímat i otisk z useknutého prstu, ale fakt, že mezi otiskem prstu a stálým heslem není z našeho pohledu příliš velký rozdíl. Snad jen v tom, že běžné heslo má řádově jednotky znaků a otisk prstů až 1000 bajtů. Jelikož se otisk nemění, bylo by jej možné na síti odchytit a následně zneužít.

Význam biometrických vlastností je spíše v umožnění přístupu k lokálním zařízením: k otevření dveří, k přístupu k PC apod. A právě kombinace přístupu pomocí otisků prstů k čipové kartě a následné využití čipové karty je již špičkovou technologií. Pomocí otisků prstů a PINu otevřeme přístup k soukromému klíči na čipové kartě a následně využijeme soukromého klíče na této kartě pomocí asymetrické kryptografie k autentizaci.

1.10.3 Shamirův algoritmus

Na obecné škole se učíme, že české korunovační klenoty jsou zajištěny několika zámky. Pro přistup ke klenotům je pak nutná přítomnost všech držitelů příslušných klíčů s jejich klíči.

Shamirův algoritmus je určen pro ochranu aktiva (šifrovacího klíče, sdíleného tajemství) tak, aby pro získání tohoto aktiva byla nutné sestavit tajem, jehož jednotlivé části jsou distribuovány mezi ***n*** uživatelů. Rozdíl oproti přístupu ke korunovačním klenotům je v tom, že pro sestavení celého tajemství se nemusí sejít všech ***n*** uživatelů, ale stačí jen libovolných ***k*** z nich (platí tedy: $0 < \boldsymbol{k} \leq \boldsymbol{n}$).



Obrázek 1.20 Shamirův algoritmus *(**k**=2)*

Prakticky to znamená, že nějaké indicie uložíme např. na ***n*** čipových karet, které rozdáme ***n*** různým držitelům. Pokud potřebujeme tajemství rekonstruovat, pak se musí sejít alespoň ***k*** držitelů se svými kartami.

Princip Shamirova algoritmu je triviální. Představme si, že mezi 5 správců chceme rozdat nějaké indicie tak, abychom vždy, když se sejdou alespoň dva, dali dohromady tajemství B.

Pak zvolíme libovolné (ale pevné) číslo A a sestavíme rovnici přímky:

$$y = Ax + B$$

Každému z pěti správců pak prozradíme souřadnice nějakého bodu na této přímce. Vždy, když se sejdou dva, tak umí spočíst B, tj. sestavit tajemství. Pro $k=3$ použijeme parabolu atd.

2 Certifikáty a certifikační autority

Vraťme se zpět k našim známým: k Aloisovi, Bohumile a Cyrilovi (Obrázek 2.1). Bohumila vygenerovala dvojici asymetrických klíčů (1); vygenerovaný veřejný klíč Vk-B poslala Aloisovi po dotěrném Cyrilovi (2). Bohumila doufá, že Alois zašifruje svou odpověď jejím veřejným klíčem, jež mu přinese Cyril. Tím Alois zamezí, aby si zprávu přečetl kdokoliv jiný než Bohumila. I kdyby Alois tuto šifrovanou zprávu poslal po Cyrilovi, tak si ji ani Cyril nepřečte.

Cyril se pomalu vleče k Aloisovi a tu mu v hlavě uzraje plán (3). Zastaví se u své kamarádky – hackerky označující se jako slečna X. Třeba mu poradí, každopádně tím nic nezkazí. Slečna X si prohlédne veřejný klíč Bohumily a okamžitě odvětí Cyrilovi, že zlomit RSA algoritmus je i nad její síly. Ale existuje přece úplně jednoduchá šance jak Cyrilovi pomoci! Vezme si od Cyrila veřejný klíč Bohumily a schová si jej pro pozdější využití (4). Nyní sama vygeneruje svou dvojici: veřejný klíč Vk-X a soukromý klíč Sk-X (5). Právě vygenerovaný veřejný klíč Vk-X dá Cyrilovi a řekneme: dones jej Aloisovi a řekni mu, že to je ten veřejný klíč, který mu posílá Bohumila. Cyril tak neprodleně učiní (6).

Nyní Alois v dobré víře, že svou odpověď šifruje veřejným klíčem Bohumily, zašifruje odpověď veřejným klíčem Vk-X a pošle ji po Cyrilovi Bohumile (7). Cyril rovnou pospíchá za slečnou X (8). Ta na něj již netrpělivě čeká. Vytrhne mu jejím veřejným klíčem Vk-X šifrovanou Aloisovu odpověď. Dešifruje ji (9) svým soukromým klíčem Sk-X a získá čistou zprávu. Tu ukáže překvapenému Cyrilovi, kterému dokonce umožní zprávu změnit v jeho prospěch. Výsledek pak šifruje veřejným klíčem Bohumily Vk-B (10) a po Cyrilovi ho pošle Bohumile (11). Nic netušící Bohumila dešifruje zprávu svým soukromým klíčem Sk-B (12).

Všichni jsou šťastní. Alois zprávu šifroval, tak jak mu kázal dobrý mrav. Bohumila obdržela zprávu od Aloise, kterou dešifrovala, tak jak měla. Cyril si nejenom mohl přečíst obsah zprávy, ale dokonce jej mohl i změnit ve svůj prospěch. A Slečna X se rovněž realizovala. Všichni jsou tudíž šťastní! Ve světě businessu bychom řekli, že všichni postupovali dle ISO 9001.

2.1 Jaká je obrana?

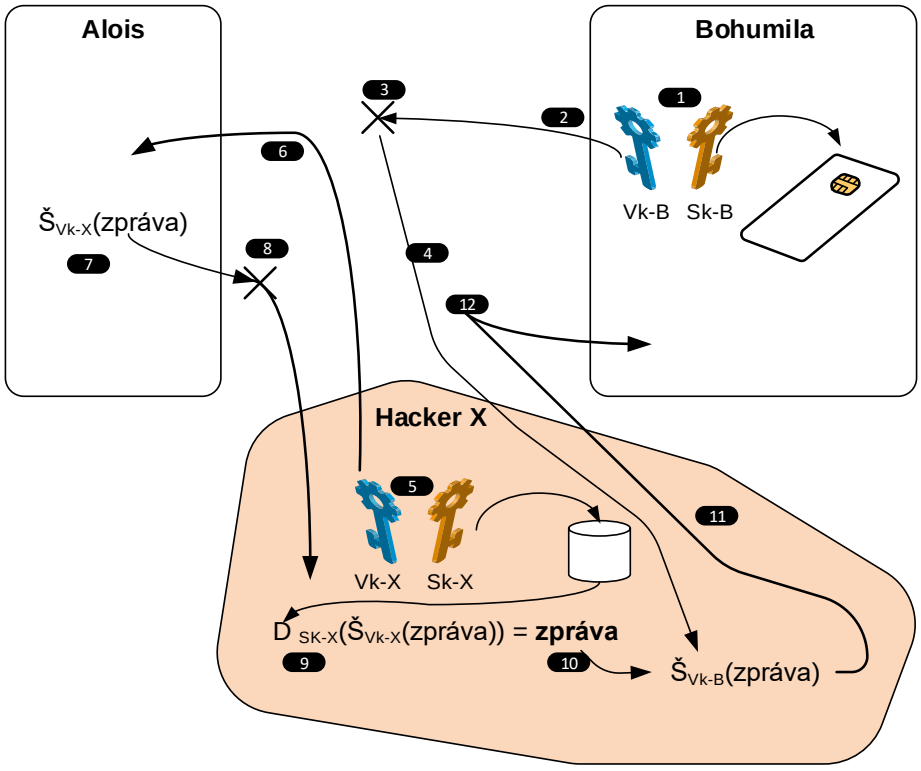
Je tedy obrana proti útokům na distribuci veřejného klíče? Před tím, než Alois použije nějaký veřejný klíč, tak by si měl obstarat nějaký důkaz, že veřejný klíč je opravdu jeho, tj. že není podvržen.

Alois má následující možnosti pro ověření pravosti Bohumilina veřejného klíče:

- Alois obdrží veřejný klíč osobně přímo od Bohumily na jejich vzájemné schůzce. Je tedy nad všechny pochybnosti jasné, že veřejný klíč je Bohumily.
- Alois si ověří, že veřejný klíč je opravdu Bohumily. Např. před prvním použitím klíče zavolá Bohumile, autentizuje ji, aby si byl jist, že telefonuje opravdu s ní a ne s nějakým útočníkem, a požádá ji, aby mu zarecitovala otisk z veřejného klíče nebo jeho část. Autentizaci provede např. na základě společně prožitého zážitku: „Jak se ti líbil ten film, na kterém jsme spolu včera byli?“. A Bohumila odpoví: „Co blbneš, vždyť včera jsme spolu byli v divadle“.
- Bohumila si nechá stvrdit nezávislou třetí stranou (např. notářem) pravost svého veřejného klíče.

2.1.1 Vlastní Bohumila odpovídající soukromý klíč?

I když si je Alois zcela jist, že veřejný klíč dostal od Bohumily, je pro něj důležité si ověřit, že Bohumila má ke svému veřejnému klíči příslušný soukromý klíč. Proč? Představme si situaci, že Bohumila má podezření, že Alois miluje kromě ní ještě její, dnes již bývalou, kamarádku Hanu. Když Alois posílá milostná psaní Bohumile, pak je šifruje veřejným klíčem Bohumily. Když posílá psaní



Obrázek 2.1 Útok na distribuci veřejného klíče Bohumily

Haně, pak je šifruje veřejným klíčem Hany. Alois si je natolik jist asymetrickou kryptografii, že dokonce po Bohumile posílá Haně Haniným veřejným klíčem šifrované zprávy. Alois Bohumile přitom tvrdí, že to je čistě obchodní věc – nic soukromého. Bohumile ani nezáleží na tom, co Alois Haně píše, to jí je vcelku jasné. Musí tomu udělat přítrž.

A tak sdělí Aloisovi, že z kryptografických důvodů přejde na nový pár veřejný/soukromý klíč. Jenže nevygeneruje novou dvojici veřejný/soukromý klíč, ale vezme Hanin veřejný klíč a předá jej Aloisovi jako by to byl její veřejný klíč. Alois si ničeho nevšimne a vesele komunikuje dál s Hanou i s Bohumilou. Bohumila oželí, že se nedozví, co ji Alois píše a po krátkém čase potká Aloise a mezi řečí mu sdělí: „Stala se nám s Hanou taková až neuvěřitelná věc. Zjistili jsme, že máme stejný veřejný klíč. To asi máme stejný i soukromý klíč? To si obě můžeme dešifrovat tvé zprávy“.

Bylo by tedy nanejvýš vhodné, aby Bohumila také podala Aloisovi důkaz o tom, že vlastní i odpovídající soukromý klíč (*private key possession*). Takovým důkazem může např. být elektronický podpis vytvořený odpovídajícím soukromým klíčem z veřejného klíče nebo z nějaké datové struktury, která veřejný klíč obsahuje. Tento typ důkazu je však možné provádět jen tehdy, když je pomocí odpovídajícího soukromého možné vytvářet elektronický podpis.

V případě asymetrických algoritmů neumožňujících elektronický podpis ale umožňujících šifrování se důkaz o vlastnictví příslušného soukromého klíče postaví na šifrování. Např. Alois šifruje Bohumile zprávu jejím veřejným klíčem a čeká, zdali ji porozuměla, tj. jestli má k dispozici odpovídající soukromý klíč.

2.1.2 Důkaz o vlastnictví soukromého klíče

Žárlivá Bohumila je obeznámená se základy kryptografie. Instaluje program Wireshark a čeká, že bude Hana posílat svůj veřejný klíč včetně důkazu o vlastnictví příslušného soukromého klíče Aloisovi. Bohumila při trošce štěstí odchytne jak veřejný klíč, tak i důkaz o vlastnictví soukromého klíče vytvořený jako elektronický podpis zasílané zprávy. Nyní může Bohumila zaslat totéž i Aloisovi (*replay attack*). Pokud by se chtěl Alois bránit proti tomuto typu útoků, pak by měl kontrolovat, jestli už v minulosti náhodou neobdržel stejný veřejný klíč, čímž by ošetřil i případ, že opravdu si (nejspíše chybou v software) obě vygenerovaly stejné klíče.

Důkaz o vlastnictví soukromého klíče má kromě kryptologických aspektů i aspekty čistě technické. Když Alois chce šifrovat za využití veřejného klíče, pak se nesmí splést ani v jednom bitu veřejného klíče. Jinak by Bohumila nedokázala zprávu dešifrovat. A pokud Alois správně verifikuje důkaz vlastnictví soukromého klíče, pak si je zpravidla jist, že správně interpretuje všechny bity veřejného klíče. Tj. ví, že pokud Bohumila nedešifruje zprávu, tak chyba není na jeho straně.

2.1.3 Generovala Bohumila svá párová data na bezpečném zařízení?

Máme důkaz o tom, že veřejný klíč patři konkrétní osobě a jiný důkaz o tom, že daná osoba k tomuto veřejnému klíči má příslušný soukromý klíč. A aby toho nebylo dost, tak máme ještě třetí důkaz: důkaz o tom, že příslušný pár klíčů byl generován a chráněn zařízením odpovídající bezpečnostní úrovně.

Soukromý klíč je aktivum, které může mít značnou hodnotu. Takové aktivum se snažíme odpovídajícím způsobem střežit. Soukromý klíč proto mnohdy udržujeme v odpovídajících zařízeních. Takovým zařízením mohou být čipové karty, USB tokeny či HSM, které samy generují pár klíčů a soukromý klíč nikdy tato zařízení neopouští. Bylo by nemilé, kdybychom např. omylem vygenerovali pár klíčů mimo toto zařízení a soukromý klíč uložili na disk. Investice do speciálního zařízení by vešla vniveč a navíc bychom naše střežené aktivum vystavili nebezpečným rizikům.

Jako důkaz, že soukromý klíč je střežen odpovídajícím zařízením zpravidla využíváme jednorázové hesla, která toto zařízení generuje společně s odpovídajícím párem klíčů. Jak je ale možné, že nám stačí jedno jednorázové heslo při generování páru klíčů? Tento důkaz nám totiž stačí pouze jednou – v okamžiku vytváření žádosti o certifikát.

2.1.4 Závěr

Pokud Alois přímo použije Bohumilin veřejný klíč, pak by měl od Bohumily obdržet:

1. Samotný veřejný klíč.
2. Důkaz o tom, že tento veřejný klíč opravdu patří Bohumile.
3. Důkaz o tom, že Bohumila má k tomuto veřejnému klíči opravdu příslušný soukromý klíč.
4. V případě, že se jedná zabezpečení dat značné ceny, tak důkaz o tom, že párová data byla generována na odpovídajícím zařízení a příslušný soukromý klíč je odpovídajícím způsobem chráněn.

Pokud si Bohumila nechá ověřit platnost svého soukromého klíče u nezávislé třetí strany (např. notáře), pak čtyři zmíněné úkony předvede notáři a notář vystaví certifikát o platnosti veřejného klíče Bohumila. Bohumila pak Aloisovi předá veřejný klíč a příslušný certifikát tohoto klíče vydaný notářem (vytištěný na papíře např. v hexadecimálním tvaru).

2.2 Certifikace veřejného klíče

Proti podvržení veřejného klíče je však praktičtější se bránit certifikací veřejného klíče nezávislou třetí stranou –

certifikační autoritou (Obrázek 2.2). Bohumila si vygeneruje dvojici veřejný/soukromý klíč, přičemž soukromý klíč si jako své tajné aktivum pečlivě uloží a střeží. Veřejný klíč nezašle Aloisovi samotný, ale až jako součást certifikátu vydaného certifikační autoritou. Avšak nepředbíhejme.

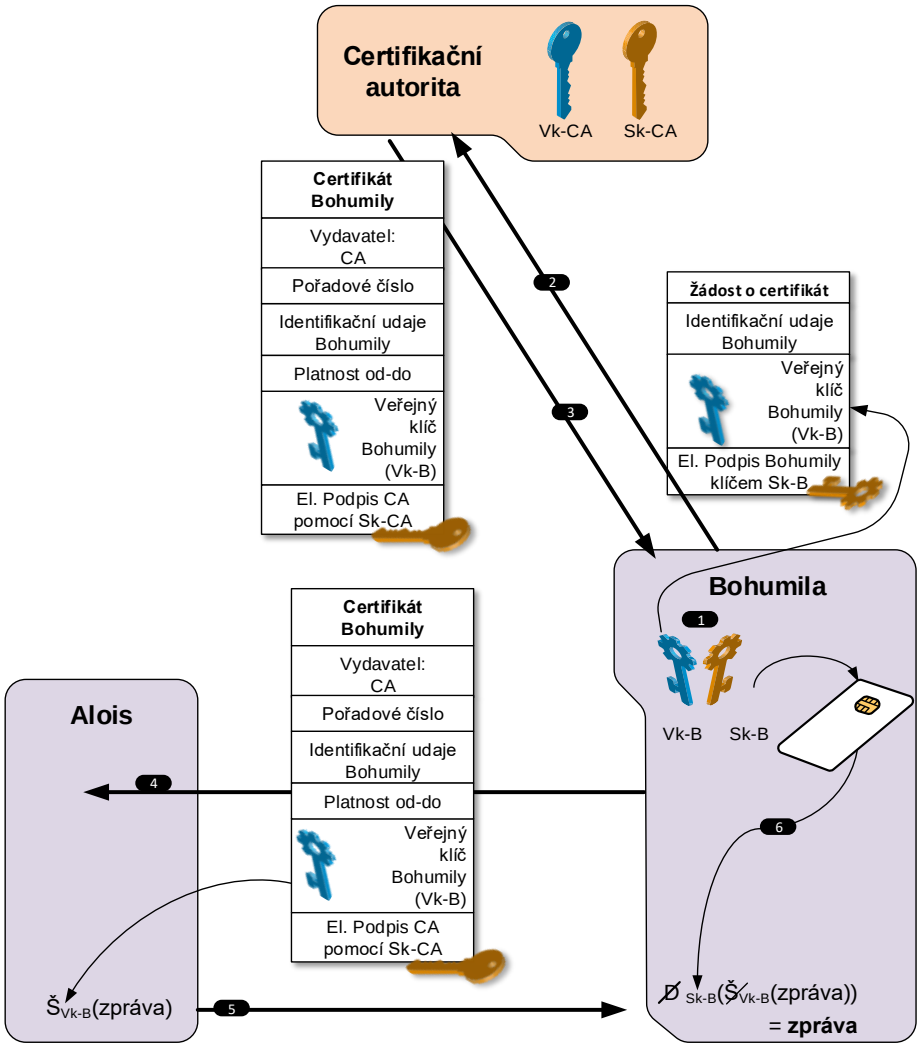
Po vygenerování dvojice klíčů (1) Bohumila sestaví strukturu „žádost o certifikát“. Tato struktura obsahuje identifikační údaje Bohumily, její veřejný klíč a případně další data, o kterých si povíme později. Tuto strukturu digitálně podepíše svým právě vygenerovaným soukromým klíčem (= důkaz o vlastnictví soukromého klíče) a výsledek předá certifikační autoritě (2). Certifikační autorita může ověřit totožnost Bohumily. V každém případě však verifikuje elektronický podpis na žádosti o certifikát, aby si CA ověřila, že Bohumila opravdu vlastní příslušný soukromý klíč. Pokud je žádost certifikační autoritou shledána v pořádku, pak certifikační autorita vystaví certifikát.

Certifikát je datová struktura obsahující veřejný klíč Bohumily a její identifikační údaje. Dále certifikát obsahuje název vydavatele certifikátu (jedinečné jméno certifikační autority), pořadové číslo vydaného certifikátu, platnost certifikátu atd. Spojení identifikačních údajů Bohumily a jejího veřejného klíče stvrzuje certifikační autorita svým elektronickým podpisem.

Bohumile je certifikační autoritou vrácen vystavený certifikát (3). Nyní již může Bohumila svůj veřejný klíč poslat i Aloisovi jako součást právě vystaveného certifikátu (4). Alois ověří tento certifikát a v případě, že je vše OK extrahuje z tohoto certifikátu veřejný klíč, který následně využije k šifrování zprávy Bohumile (5). Bohumila pak pomocí svého soukromého klíče zprávu dešifruje (6) a získá tak původní zprávu.

Co Alois na certifikátu Bohumily ověřuje? Později si povíme, že ověřování certifikátu je docela komplikovanou procedurou. V tomto okamžiku jen připomeňme, že Alois musí ověřit, zdali byl certifikát Bohumily vydán pro něj důvěryhodnou certifikační autoritou (jejíž identifikace je v položce vydavatel certifikátu). Jelikož certifikát je digitálně podepsaná struktura, tak Alois musí ověřit elektronický podpis na Bohumilině certifikátu.

Na obrázku Obrázek 2.4 je znázorněna vazba mezi certifikátem Bohumily a certifikátem CA, která Bohumile certifikát vydala. Alois si musí nalézt příslušný certifikát certifikační autority ve svém úložišti certifikátů důvěryhodných certifikačních autorit nebo jiným mechanismem (např. pomocí rozšíření Přístup k informacím úřadu). Postupuje tak, že nejprve prohledá své úložiště důvěryhod-

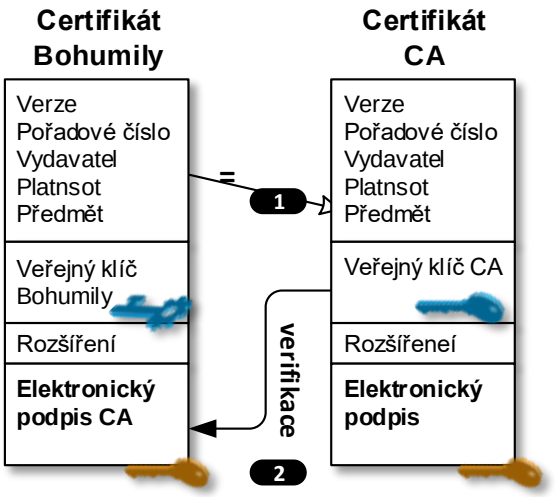


Obrázek 2.2 Jak certifikát funguje

ných certifikátů a vyhledá v něm certifikát certifikační autority, která vydala certifikát Bohumile. Ten pozná podle toho, že má identickou položku Předmět s položkou Vydavatel v Bohumilině certifikátu (1). Poté vyextrahuje veřejný klíč z certifikátu certifikační autority a pomocí něj verifikuje elektronický podpis v Bohumilině certifikátu (2).

Certifikáty certifikačních autorit jsou podobné uživatelským certifikátům. Svůj certifikát si certifikační autorita může nechat vydat:

- U jiné certifikační autority, pak certifikát této CA má různé položky Vydavatel a Předmět. Takový certifikát CA nazýváme křížovým certifikátem (*cross certificate*).



Obrázek 2.4 Zjednodušené ověření (verifikace) certifikátu Bohumily

- Může jej vydat sama, tj. i podepisuje si jej sama svým soukromým klíčem, pak certifikát této CA má shodné položky Vydavatel a Předmět. Takový certifikát označujeme jako kořenový (*self signed certificate*).

2.2.1 Achillova pata certifikátu

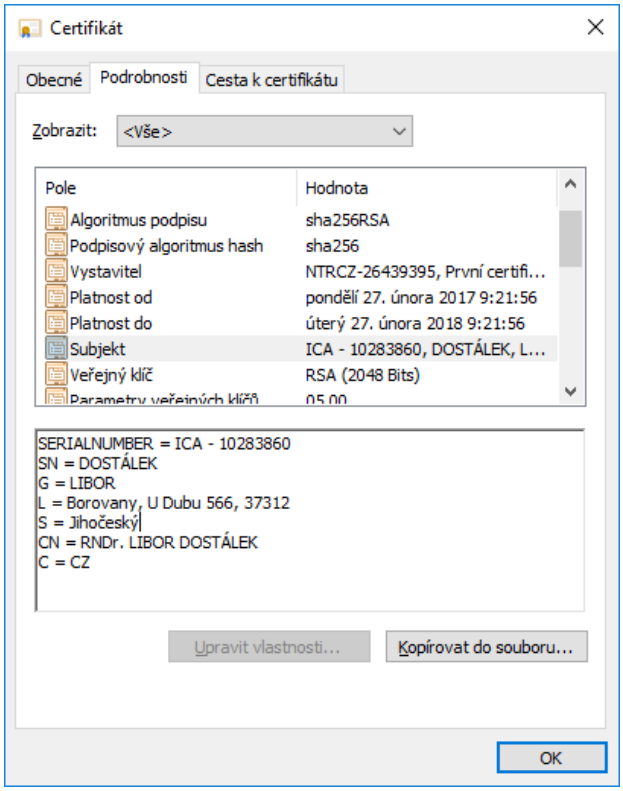
Certifikát je veřejná listina. Cílem držitele certifikátu je maximálně certifikát rozšiřovat. Nepříjemností by ale bylo, kdyby certifikační autorita vystavila nějakému uživateli certifikát pro konkrétní veřejný klíč a záhy by se objevil jiný uživatel s žádostí o certifikaci téhož klíče. Protože, když tito uživatelé mají stejný veřejný klíč, tak si navzájem znají i soukromé klíče.

Certifikační autorita by každopádně měla sledovat, zdali již stejný veřejný klíč necertifikovala. A další žádost o certifikaci téhož klíče odmítnout. Zde je vidět jak praktické je doplňovat žádost o certifikát důkazem o vlastnictví soukromého klíče. Bez tohoto důkazu by mohl o certifikaci již certifikovaného veřejného klíče žádat každý, komu se nějaký certifikát dostane do ruky.

Generování shodných párových dat se zamezujeme využíváním kvalitních generátorů náhodných čísel při generování dvojice veřejný/soukromý klíč. Používají se tzv. pravé generátory náhodných čísel (*true random*). Je pak nepravděpodobné, že by si dva různí uživatelé vygenerovali stejná párová data. Ale chybou software se může stát, že generátor náhodných čísel čísla negeneruje zcela náhodně. Jsou popsány i případy, kdy chyba generátoru náhodných čísel byla navozena uměle pomocí tzv. útoků postranními kanály.

2.3 Certifikát

Certifikát se často přirovnává k občanskému průkazu či pasu. Zatímco občanský průkaz se vydává v tištěné po-



Obrázek 2.3 Zobrazení obsahu certifikátu ve Windows

době, certifikát je digitálně podepsanou datovou strukturou, jejíž základní součástí je veřejný klíč držitele certifikátu.

Můžeme i porovnat jednotlivé položky občanského průkazu a certifikátu (později se dozvíme, že toto porovnání je základem práce zaměstnance registrační autority při ověřování totožnosti žadatele o certifikát):

Položka certifikátu	**Položka občanského průkazu**
Verze (*Version*)	Verze formátu občanského průkazu (knížka, karta apod..)
Pořadové číslo (*Serial numer*)	Číslo občanského průkazu
Algoritmus podpisu (*Signature Algorithm*)	Způsob podpisu úředníka, typy ochranných prvků
Vydavatel (*Issuer*)	Vydal
Platnost (*Valididty*)	platnost
Předmět: jméno, adresa, (*Subject*)	Jméno a adresa
Veřejný klíč (*Subject public key*)	-
-	Fotografie
Rozšíření certifikátu (*Extension*)	Nepovinné údaje
Elektronický podpis (*Digital signature*)	Ochranné prvky

Máme několik norem, které definují strukturu certifikátu (X.509, EDI, EMV apod.). V Internetu se vychází ze standardu X.509 verze 3, který vydal ITU. Pro potřeby Internetu je vytvořen internetový profile standardu X.509 v příslušném RFC. Aktuálním internetovým profilem certifikátu je dnes standard RFC-5280.

Nyní se zastavíme u jednotlivých položek certifikátu.

2.3.1 Verze certifikátu

Verze certifikátu souvisí s tím, je-li certifikát odvozen od normy X.509 verze 1, 2 nebo 3[*)]. Položka *Version* má v případě verze jedna hodnotu nula, v případě verze 2 hodnotu 1 a v případě verze 3 hodnou 2. Dnes se zásadně používají pouze certifikáty verze 3.

2.3.2 Pořadové číslo certifikátu

Pořadové číslo certifikátu (*Serial Number*) je definováno jako celé kladné číslo, které musí být jednoznačné v rámci konkrétní certifikační autority. Tj. certifikační autorita nesmí vydat dva certifikáty, které by měly stejné pořadové číslo. Dvojice položek *Serial Number* + *Issuer* jednoznačně identifikují certifikát.

2.3.3 Algoritmus podpisu

Položka Algoritmus podpisu (*Signature Algorithm*) specifikuje algoritmy použité CA pro vytvoření elektronického podpisu certifikátu. Tato položka vždy specifikuje dvojici algoritmů:

- Jeden pro výpočet otisku (hash)
- Druhým algoritmem je asymetrický algoritmus, kterým je kontrolní součet šifrován.

2.3.4 Platnost

Položka Platnost (*Validity*), určující platnost certifikátu od (*Not Before*) do (*Not After*).

Častou otázkou je, proč je omezena doba platnosti certifikátu. Důvody jsou dva:

- **Organizační**, tj. aplikace má určitou životnost. Bezesporu je i obchodně zajímavé vydávat certifikáty častěji atd.
- **Bezpečnostní**, což je pádnější důvod. Životnost certifikátu by měla být výrazně kratší než doba nutná k prolomení certifikovaného veřejného klíče. To je ovšem problém zejména u certifikátů certifikačních autorit, které by měly být vydávány na dobu alespoň pětkrát delší, než je životnost uživatelských certifikátů (při kratší době se silně zvyšuje režie obnovování uživatelských certifikátů).

Je vždy znovu a znovu třeba zdůrazňovat, že certifikát po vypršení doby platnosti není k nepotřebě a tudíž nemůže být bezstarostně zahozen. Pomocí soukromého klíče příslušejícího k veřejnému klíči uvedenému v certifikátu po vypršení doby platnosti pouze nepodepisujeme nové zprávy. Avšak k ověření elektronického podpisu zpráv vytvořených v době platnosti certifikátu budeme v budoucnu vždy potřebovat i prošlé certifikáty. A budeme je potřebovat tak dlouho, dokud se ověřování bude provádět.

Obdobně, pokud si budeme archivovat zprávy zašifrované veřejným klíčem z certifikátu, pak k jejich dešifrování opět budeme potřebovat soukromý klíč k certifikátu, který byl v době vytvoření zprávy platný. Při zpracování zprávy je proto praktické zprávu dešifrovat a před ukládáním do archivu ji případně znovu šifrovat, ale tentokrát šifrovacím klíčem archivu.

2.3.5 Položky Vydavatel a Předmět

Položka Vydavatel (*Issuer*) specifikuje toho, kdo certifikát vydal, tj. certifikační autoritu. Položka Předmět (*Subject*) specifikuje držitele certifikátu.

Obě položky Vydavatel i Předmět používají stejný datový formát označovaný jako jedinečné jméno (*Distiguished Name*).

2.3.5.1 Jedinečné jméno

Jedinečné jméno (*Distiguished Name*) bylo původně zavedeno v normách ITU řady X.500, konkrétně v normě X.501. Cílem norem řady X.500 je vytvořit celosvětovou adresářovou strukturou. Adresářem se přitom nerozumí adresář souborů, ale adresář jako seznam adres v telefonním seznamu. Cílem je tak vytvořit celosvětovou obdobu telefonního seznamu. Jeden záznam v takovém seznamu pak odpovídá jedinečnému jménu.

Jedinečné jméno by v takovém seznamu pak bylo tvořeno dílčími informacemi o tomto subjektu: názvem země, názvem telefonní společnosti, telefonním obvodě, jméně, adrese a konečně telefonním číslem. Takovou konkrétní dílčí informaci, ze které se skládá jedinečné jméno, nazýváme relativním jedinečným jménem (*Relative Distinguished Name*).

Jedinečné jméno je tak tvořeno posloupností relativních jedinečných jmen. Přitom v jedinečném jméně se mohou i relativní jedinečná jména opakovat (např. s jinou hodnotou).

Samotné relativní jedinečné jméno je množina atributů. Atribut je pak dvojice tvořená identifikátorem objektu (např. Country, Organization, Common Name apod.) a

[*)] Norma X.509 verze 4 již strukturu certifikátu nemění, proto též využívá v položce *Version* hodnotu 2.

hodnotou (např. CZ). Relativní jedinečné jméno zapisujeme např.:

`Common Name=Libor Dostalek`

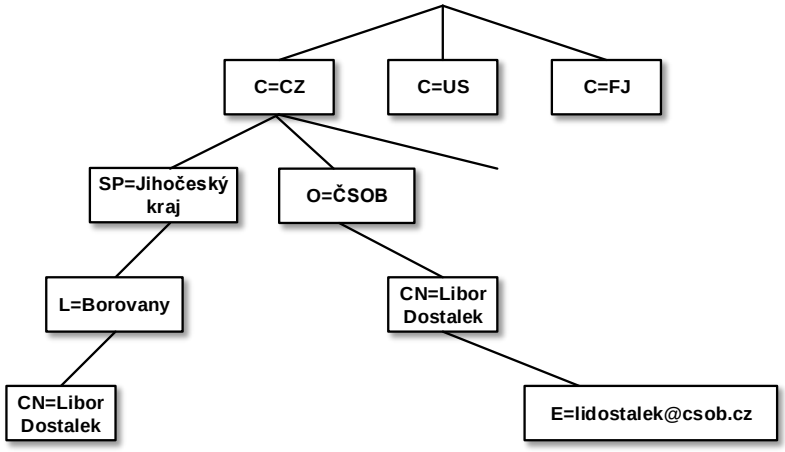
Jedinečné jméno popisující jedince je pak sekvencí relativních jedinečných jmen. Např.:

```
CommonName=Libor Dostalek, Organia-
tion=ČSOB, Country=CZ
```

Tento zápis se často zkracuje pomocí zkratek pro identifikátory objektů relativních jedinečných jmen:

`CN=Libor Dostalek, O=ČSOB ,C=CZ`

I když relativní jedinečné jméno je množinou atributů, v praxi bývá tato množina jednoprvková, tj. obsahuje jen jeden atribut (jednu dvojici identifikátor + hodnota). Jedinečná jména jsou tvořena vždy větví ve stromu relativních jedinečných jmen (Obrázek 2.5).



Obrázek 2.5 Hierarchická struktura relativních jedinečných jmen

Zajímavé je, že Libor Dostalek může být ve struktuře uveden mnoha způsoby (pokaždé se jedná o jiné jedinečné jméno!).

Např.:

- Jako obyvatel ČR, konkrétně Borovan:
  `CN=Libor Dostalek, L=Borovany, SP=Jihočeský kraj C=CZ`
- Jako zaměstnanec ČSOB
  `CN=Libor Dostalek, O=ČSOB, C=CZ`

Uvedli jsme, že relativní jedinečné jméno se zpravidla skládá z jedné dvojice identifikátor objektu a hodnota (např. `C=CZ`). Jedinečné jméno je posloupnost tvořená posloupností těchto dvojic. Jednotlivé prvky této posloupnosti se oddělují čárkou.

Jenže jak se zapíše, když by teoreticky bylo relativní jedinečné jméno tvořeno dvěma dvojicemi? Takové dvojice se oddělí znakem plus. Např.:

`OU=ČSOB+CN=Libor Dostálek, C=CZ`

specifikuje jedinečné jméno tvořeno dvěma relativními jedinečnými jmény. Přičemž první obsahuje dvě dvojice (dva atributy `OU=ČSOB` a `CN=Libor Dostálek`).

Pokud se blíže zajímáte o to, jak zapsat jedinečné jméno jako textový řetěze, pak doporučuji si prohlédnout krátkou normu RFC-4510, která tuto problematiku řeší pro LDAP.

Pomocí jedinečného jména specifikujeme osobu, systém či obecně nějakou entitu. Zajímavá situace je u jmen fyzických osob. Různé země mají své zvyklosti pro pojmenování svých občanů, proto konkrétní využití jednotlivých atributů závisí na certifikační politice konkrétní certifikační autority.

Uživatel uvede své jedinečné jméno do žádosti o certifikát a certifikační autorita toto jedinečné jméno zkopíruje do certifikátu. Obecně by certifikační autorita neměla modifikovat jedinečné jméno při kopírování z žádosti o certifikát do certifikátu. Výjimkou jsou pouze atributy *DNQualifier* a *SerialNumber*, ty naopak bude certifikační autorita s největší pravděpodobností do certifikátu doplňovat. Tyto dva atributy totiž slouží k rozlišení dvou různých osob, které by jinak měly stejné jedinečné jméno.

Rozdíl mezi *DNQualifier* a *SerialNuber* je v tom, že atributem *DNQualifier* rozlišujeme osoby, které by měly shodou okolností jinak stejný předmět. Kdežto atributem *SerialNumber* můžeme rozlišit dva certifikáty téže osoby. Např. má-li osoba vydány dva podepisovací certifikáty jiné bezpečnostní úrovně: jeden má soukromý klíč např. na disku a druhý na čipové kartě. Avšak používání *DNQualifier* a *SerialNumber* není ustálené. Certifikační autority často *DNQualifier* nepoužívají a atribut SerialNumber pak použijí pro rozlišení osob. Konkrétní význam atributů *DNQualifier* a *SerialNuber* tak musíme hledat v příslušné certifikační politice konkrétní certifikační autority.

Dalším častým zvykem certifikačních autorit je přesunutí atributu *E-mail Address* z předmětu žádosti o certifikát do příslušného rozšíření certifikátu, protože dnešní standardy přímo vyzývají certifikační autority, aby atribut *E-mail Address* neuváděli v předmětu certifikátů.

Přehled atributů relativních jedinečných jmen používaných PKI:

Atribut	Zkratka	Význam
Common Name	CN	Název objektu, pod kterým je místně znám. Např. u osob to může být jméno a příjmení. U serverů pak jejich DNS-jméno apod.
Surname	SN	Příjmení

Country	C	Stát podle ISO 3166, tj. podle stejné normy, jaká se používá pro top level domény DNS (CZ = Česká republika, SK = Slovensko, FJ = Fidži...)
Locality	L	Lokalita (např. město)
State or Province	SP or ST	Nižší organizační jednotka státu. Např. kraj či spolková země.
Organization	O	Název firmy
Organizational Unit	OU	Organizační jednotka (např. oddělení)
Title	T	Pozice (např. hejtman, jednatel společnosti, ředitel apod.)
Name		Jméno
Given Name	G	Jméno
Initials		Iniciály,
Generation Qualifier		Např. „Jr." či „IV" pro Karel IV.
DNQualifier		Slouží k rozlišení různých certifikovaných objektů, kterým by jinak vycházel stejný předmět.
Serial Number*)		Slouží k rozlišení různých certifikovaných objektů, kterým by jinak vycházel Stejný předmět. (Nezaměňovat se sériovým číslem certifikátu.)
Pseudonym	P	Pseudonym
E-mail Address	E	Adresa elektronické pošty (dle RFC-822).
Domain Component	DC	Jednotlivé řetězce z doménového jména (např. domény Windows 2000/2003). Např. domain Controller info.pvt.net je DC = www, DC = cpress, DC = cz. Pozor (!) tento atribut Nemá přímou souvislost s DNSname – jedná se o jiný prostor jmen.

2.3.5.2 Vydavatel certifikátu

Položka Vydavatel (*Issuer)* obsahuje jedinečné jméno certifikační autority. Je třeba, aby certifikační autorita měla jednoznačnou identifikaci (jedinečné jméno) v rámci všech certifikačních autorit.

Útočník bude mít snahu vytvořit certifikační autoritu stejného jména, ale za využití své podvržené dvojice veřejný/soukromý klíč. Zejména pokud naše certifikační autorita používá kořenový certifikát, tak na jeho distribuci musíme být obzvláště opatrní.

2.3.5.3 Předmět certifikátu

Pokud používáme certifikáty dle X.509 verze 3, pak předmět certifikátu musí být jedinečný v rámci všech objektů certifikovaných danou certifikační autoritou. Tj. certifikační autorita nesmí vydat dvěma různým osobám certifikát se stejným předmětem. Na druhou stranu je velice praktické, že certifikační autorita může vydávat jedné osobě certifikáty stále se stejným předmětem (stejným jedinečným jménem). Tj. Václav Vopička může mít více různých certifikátů se stejným předmětem, protože se jedná o stejného Václava Vopičku. Ale jeho jmenovec, který se jen shodou okolností také jmenuje Václav Vopička, musí mít jiný předmět. Může mít např. jinou lokalitu (město), ale kdyby všechny ostatní údaje byly stejné, pak CA použije k rozlišení jedinečné jméno *dnQualifier* či jedinečné jméno *serialNumber* (nezaměňovat s číslem certifikátu – to musí být v každém případě různé).

V případě, že by pro dva různé objekty vycházel stejný předmět, pro rozlišení objektů se použije atribut *dnQualifier* nebo *serialNumber*.

V předmětu certifikátu zpravidla využíváme širší paletu atributů jedinečných jmen než u jedinečného jména vystavitele, kde bychom měli být střídmí, i když software má podporovat nejrůznější atributy.

Mnohé identifikační údaje, které se „nevejdou" do předmětu certifikátu, je možno uložit do rozšíření certifikátu.

Zajímavostí je, že předmět certifikátu může být i prázdný (prázdná sekvence relativních jedinečných jmen). V takovém případě ale certifikát musí povinně obsahovat rozšíření Alternativní jméno předmětu, které musí být označeno jako závažné.

2.3.6 Veřejný klíč

Položka Veřejný klíč (*Subject Public Key*) je sekvencí dvou informací: identifikátorem algoritmu, pro který je veřejný klíč určen, a samotným veřejným klíčem.

Zmíněný algoritmus na rozdíl od položky Algoritmus podpisu (*Signature Algorithm*) specifikuje algoritmus, pro který je určen certifikovaný veřejný klíč.

Pokud si např. Bohumila chce certifikovat své veřejné Diffie-Hellmanovo číslo, pak položka Veřejný klíč obsahuje identifikátor pro Diffie-Hellmanův algoritmus a Bohumilino veřejné Diffie-Hellmanovo číslo. Certifikační autorita stvrzující svým podpisem platnost Bohumilina certifikátu použije pro elektronický podpis certifikátu např. algoritmus RSA. Certifikační autorita pak do položky Algoritmus podpisu vyplní identifikaci algoritmů použitých pro podpis samotného certifikátu (např. *RSA with SHA-1*).

2.4 Rozšíření certifikátu

To, co se nevešlo do předchozích položek certifikátu, se snažíme uložit do některého z rozšíření. Neměli bychom to však přehánět. Zásada je taková, že do certifikátu vkládáme informace týkající se identifikace držitele certifikátu. Někteří IT architekti se snaží do certifikátu vyznačit i role držitele certifikátu v aplikacích včetně přístupových práv. K tomuto účelu ale slouží atributové certifikáty.

I když rozšíření certifikátu je definováno zcela obecně, je i u něj potíž (podobně jako u atributů předmětu certifikátu) spočívající v tom, že aplikace některým rozšířením nebude rozumět – nebude vědět, k čemu toto konkrétní

rozšíření slouží. Tento problém řeší položka závažnost rozšíření *(critical)*. Tato položka sděluje, je-li rozšíření závažné či nikoliv. V případě, že je položka závažnost nastavena na TRUE, je rozšíření označeno jako závažné.

Software pracující s certifikátem musí rozumět všem závažným rozšířením – musí si být vědom závažnosti informací v nich uvedených. V případě, že některé z rozšíření v certifikátu je označeno jako závažné a software neví, k čemu toto rozšíření slouží, musí celý certifikát odmítnout.

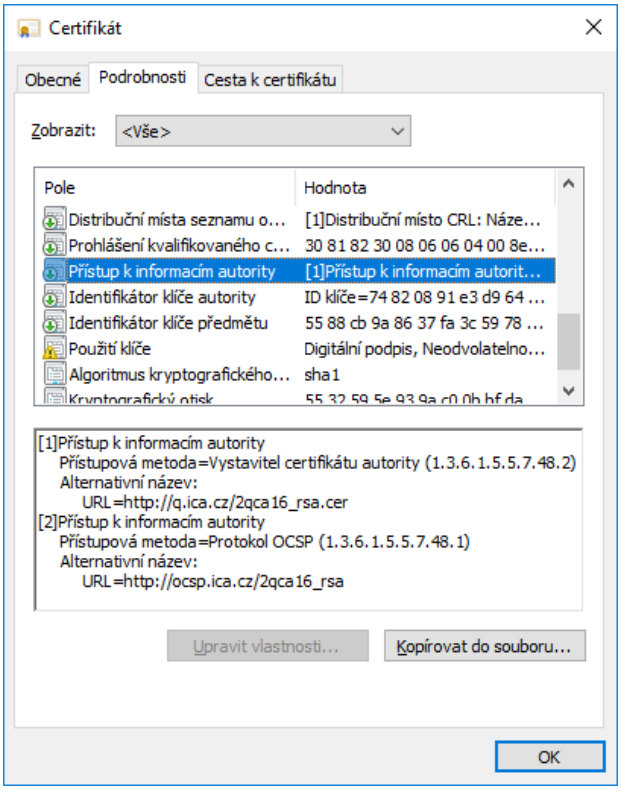
RFC-5280 specifikuje standardní rozšíření uvedené v následující tabulce. Zajímavé je, že ne každé z těchto standardních rozšíření musí být podporována konkrétními aplikacemi. Některá dokonce RFC-5280 vůbec nedoporučuje používat.

Tabulka 2.1 Některá rozšíření certifikátů

	Rozšíření certifikátu	**Certifikáty CA**	**Certifikáty koncových uživatelů**
Standardní internetová rozšíření	Identifikátor klíče úřadu *(Authority Key Identifier)*	Povinné ve všech certifikátech, které nejsou kořenové Nesmí být označeno jako závažné	
	Identifikátor klíče předmětu *(Subject Key Identifier)*	Povinné; nesmí být závažné	Mělo by být; nesmí být závažné
	Použití klíče *(Key Usage)*	Povinné v certifikátech, pomocí kterých se verifikuje elektronický podpis certifikátů a CRL; mělo by být závažné	
	Rozšířené použití klíče *(Extended Key usaje)*	Je povinné pro některé certifikáty (TSA, DVCS apod.)	
	Platnost soukromého klíče *(Private Key Usage Period)*	Nemělo by se používat	
	Certifikační politiky, též někdy Zásady certifikátu *(Certificate Policies)*	Volitelné	
	Mapování zásad (*Policy Mappings)*	Jestliže se použije, pak nesmí být závažné	-
	Subject Directory Attributes	Jestliže se použije, pak nesmí být závažné	
	Alternativní jméno předmětu *(Subject Alternative Name)*	Certifikát CA nemůže mít prázdný předmět, proto toto rozšíření je volitelné a nemusí být závažné	Jen v případě, že certifikát má prázdný předmět, tak musí být použito a navíc označeno jako závažné
	Alternativní jméno úřadu *(Issuer Alternative Name)*	Nemělo by být závažné, aplikace jej nemusí rozeznávat	
	Základní omezení *(Basic Constraints)*	Musí být použito, a to jako závažné	-
	Omezení jmen *(Name Constraints)*	Je-li použito, pak jako závažné	-
	Omezení politik *(Policy Constraints)*	Je-li použito, může i nemusí být závažné	-
	Distribuční místa seznamu odvolaných certifikátů *(CRL Distribution Points)*	Nemělo by být závažné	
	Inhibit Any-Policy	Je-li použito, pak jako závažné	-
	Nejčerstvější seznam CRL (*Freshest CRL*)	Nesmí být závažné	
Privátní internetová rozšíření	Přístup k informacím úřadu *(Authority Information Access)*	Nesmí být závažné	
	Přístup k informacím předmětu *(Subject Information Access)*	Nesmí být závažné	
Kvalifikované certifikáty	Biometrické informace *(Biometric Information)*	Nesmí být závažné	
	Qualified Certificate Statements	Může i nemusí být závažné	
Microsoft	Název šablony certifikátu *(Certificate template name)*		

2.5 Průvodce některými rozšířeními certifikátu

Ve zbytku této kapitoly se zmíníme o některých rozšířeních certifikátu. Hlavním cílem této kapitoly je poskytnout čtenáři základ, aby byl schopen porozumět těmto rozšířením, když si zobrazí obsah certifikátu např. v MS Windows. Takový výpis je např. na obrázku Obrázek 2.6, kde závažná rozšíření jsou označena vykřičníkem ve žlutém trojúhelníčku, kdežto ostatní rozšíření jsou označena zelenou šipkou v bílém terčíku. V



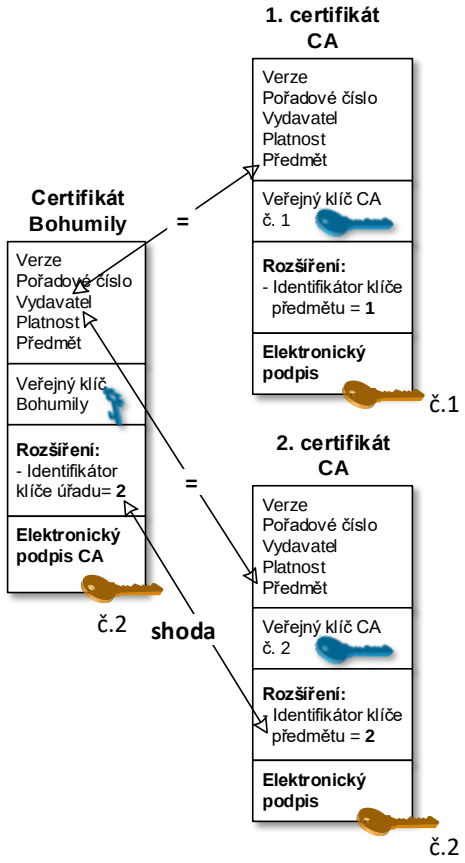
Obrázek 2.6 Rozšíření certifikátu

2.5.1 Identifikátor klíče předmětu a Identifikátor klíče úřadu

Jak jsme se již zmínili, táž osoba může mít vydáno více certifikátů. V tomto smyslu je osobou i sama certifikační autorita. Co kdybychom tedy měli např. certifikační autoritu, která by měla dva certifikáty se stejným předmětem, ale s různými veřejnými klíči?

Pak pokud by Alois chtěl verifikovat certifikát Bohumily, pak by narazil. Nevěděl by, který z certifikátů certifikační autority zvolit pro verifikaci certifikátu Bohumily. A pokud si řeknete, že je to jednoduché. Že nejprve použije jeden, a když verifikace selže, tak druhý. A co když selže i druhý certifikát CA? Co to znamená? Je certifikát Bohumily podvržen nebo naše milá certifikační autorita má ještě třetí certifikát, který je až vhodný pro verifikaci certifikátu Bohumily?

A právě rozšíření Identifikátor klíče předmětu s rozšířením Identifikátor klíče úřadu nám řeší tento problém. Certifikační autorita si označí jednotlivé své veřejné klíče (např. je očísluje ale praktičtější je klíče identifikovat otiskem z nich). Do svého certifikátu pak certifikační autorita doplní rozšíření Identifikátor klíče předmětu, do kterého uvede označení svého veřejného klíče.



Obrázek 2.7 Alois nalezne správný certifikát CA podle shody v obsahu rozšíření Authority Key Identifier a Subject Key Identifier

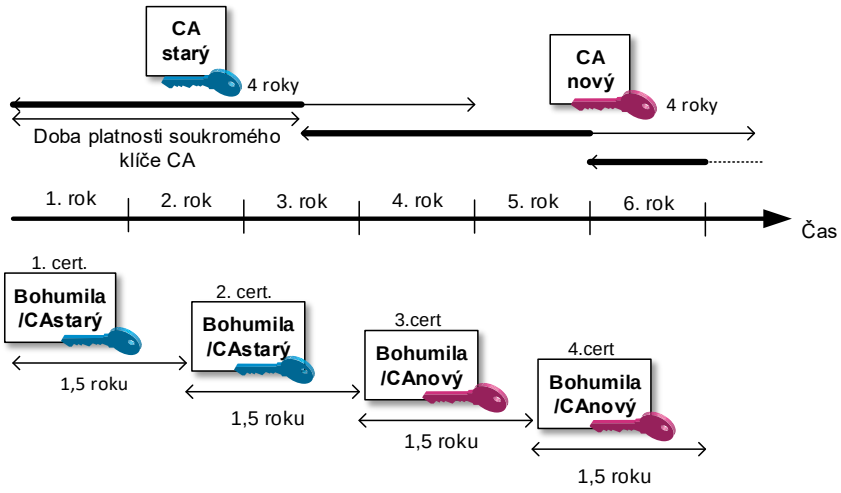
Pokud certifikační autorita vydává certifikát svým uživatelům, tak do každého vydaného certifikátu uvede rozšíření Identifikátor klíče úřadu, do kterého vloží označení veřejného klíče CA, který se má použít pro verifikaci tohoto certifikátu.

Vraťme se k Aloisovi. Když Alois prohledává své úložiště důvěryhodných certifikačních autorit, aby nalezl veřejný klíč pro verifikaci certifikátu Bohumily, pak vždy, když vyhledá certifikát certifikační

autority, který má Předmět shodný s položkou Vydavatel Bohumilina certifikátu, tak navíc zkontroluje, jestli se shoduje obsah položky identifikátor klíče úřadu v Bohumilině certifikátu s obsahem položky Identifikátor klíče předmětu v příslušném certifikátu certifikační autority.

Jinými slovy: dvojice rozšíření Identifikátor klíče předmětu a rozšíření identifikátor klíče úřadu slouží k identifikaci klíče certifikační autority, kterým byl certifikát podepsán. To je důležité zejména v případě, že certifikační autorita má více dvojic veřejný/soukromý klíč. Pokud vám připadá zbytečné, aby certifikační autorita měla více dvojic klíčů, jen jste si neuvědomili, že i certifikát certifikační autority má svou dobu platnosti. Tj. i certifikáty certifikačních autorit je třeba obnovovat. Po jistou dobu tak má certifikační autorita certifikáty dva: starý a nový (později uvidíme, že správně by měla mít po tuto dobu ještě další dva starý-nový a nový-starý).

Rozšíření Identifikátor klíče předmětu může být užitečné i v případě certifikátů koncových uživatelů. Obsahem tohoto rozšíření lze totiž efektivně identifikovat certifikát. V dalších kapitolách se pak setkáme s tím, že certifikát je buď identifikován dvojicí Vydavatel + Pořadové číslo nebo právě obsahem rozšíření Identifikátor klíče předmětu, které má velkou výhodu, že mívá pevnou délku.



Obrázek 2.8 Platnost soukromého klíče

2.5.2 Platnost soukromého klíče

Toto rozšíření umožňuje vyznačit kratší dobu platnosti soukromého klíče než je doba platnosti celého certifikátu. Což umožňuje takto označeným klíčem digitálně podepisovat např. 1 rok, kdežto verifikovat můžeme tento podpis bez potíží např. 5 let.

Zastavme se nyní u trochu jiné otázky: proč by při obnově certifikátu certifikační autority měly po jistou dobu platit oba certifikáty.

Představme si, jak pracuje Bohumilina oblíbená certifikační autorita (Obrázek 2.8), jejíž certifikáty mají platnost 4 roky a svým uživatelům (např. Bohumile) vystavuje certifikáty nejvýše na 1,5 roku.

V horní části obrázku je znázorněna platnost certifikátů certifikační autority a ve spodní části, pak platnost uživatelských certifikátů.

Zaměřme se na obrázek. Certifikační autorita vydá Bohumile první certifikát. Když se blíží vypršení tohoto certifikátu, tak certifikační autorita Bohumile vydá druhý certifikát. Oba dva certifikáty se ověřují certifikátem certifikační autority CA-starý (*CAold*). Když se blíží vypršení platnosti druhého Bohumilina certifikátu, tak si Bohumila chce opět obnovit svůj certifikát. Může ji certifikační autorita vydat certifikát, který se verifikuje certifikátem CA-starý. Nemůže! Proč? Pokud by to totiž udělala, pak po jednom roce platnosti Bohumilina certifikátu by byl Bohumilin certifikát sice formálně platný, ale jeho verifikace by selhala, jelikož certifikát CA-starý by již byl neplatný.

Závěr je tedy takový, že certifikační autorita si musí obnovit svůj certifikát nejpozději tak dlouho před vypršením svého starého certifikátu, na jak dlouho vydává certifikáty svým uživatelům. CA nemůže totiž vydat uživateli certifikát podepsaný klíčem CA, který by v době platnosti vydaného certifikátu vypršel. Tj. nastala by situace, že uživatel má sice platný certifikát, který je však podepsán neplatným (expirovaným) certifikátem.

CA tak má poměrně dlouhou dobu dva certifikáty, které se překrývají. Někteří uživatelé mají svůj certifikát podepsán „starým“ certifikátem CA a jiní „novým“ certifikátem CA. Oba certifikáty CA budou mít stejný předmět. Budou se lišit pořadovým číslem a veřejným klíčem. V certifikátu uživatele je v položce Vydavatel *(Issuer)* uveden předmět z certifikátu CA, kterým je certifikát uživatele podepsán. A ten je pro nový i starý certifikát certifikační autority stejný. A opět jsme u problému, který se

přece řeší pomocí v předchozím paragrafu zmíněného rozšíření Identifikátor klíče úřadu a Identifikátor klíče předmětu.

V horní části obrázku Obrázek 2.8 je vyznačena doba „Doba platnosti soukromého klíče“. Po tuto dobu je využíván klíč certifikační autority k podpisování vydávaných certifikátů. Po uplynutí této doby začne certifikační autorita využívat nový certifikát. Soukromý klíč příslušející starému certifikátu certifikační autority je po uplynutí této doby dobré zlikvidovat.

Dobu platnosti soukromého klíče je též možné uvést do stejnojmenného rozšíření, které však není doporučeno využívat. Toto rozšíření muže též omezit platnost soukromého klíče i na počátku platnosti certifikátu.

2.5.3 Použití klíče

Pomocí tohoto rozšíření lze omezit způsob použití veřejného klíče obsaženého v certifikátu, tj. omezit použití certifikátu. Toto rozšíření obsahuje bitový řetězec. Každý bit z řetězce pak odpovídá konkrétnímu způsobu použití certifikátu. Je-li příslušný bit nastaven na TRUE, pak je certifikát k danému použití možno používat. (Pokud se daný bit v řetězci nevyskytuje, pak se předpokládá jeho nastavení na TRUE.)

Význam jednotlivých bitů:

- ***Digital Signature*** – certifikát je určen k elektronickému podpisu dat (jako algoritmu). Nastavení tohoto bitu **ne**opravňuje k:
    - Ověřování pravosti (či jestli chcete nepopiratelné odpovědnost – k tomu je určen až následující bit - *Non Repudation*. To vás asi překvapilo.

  Zřejmě si říkáte, k čemu tedy může takový certifikát sloužit. **Může** sloužit k:
    - Autentizaci uživatelů
    - K ověřování integrity dat.
- ***Non Repudation*** – certifikát je určen k ověřování pravosti či jestli chcete nepopiratelné odpovědnosti. Názvy bitu *Digital Signature* a bitu *Non Repudation* jsou naprosto zavádějící. Bit *Digital Signature* signalizuje libovolné použití, které vychází z algoritmu elektronickýho podpisu. Nikoliv tedy z konkrétního využití pro ověřování pravosti v případě elektronickýho podpisu. Bit *Non Repudiation* pak signalizuje konkrétní využití algoritmu elektronickýho podpisu jako ověřování pravosti. Takže pokud chceme např. využít certifikát k verifikaci elektronickýho podpisu listiny, který má nahrazovat rukou psaný podpis, pak musí být nastaveny oba bity. První signalizuje podporu algoritmu a druhý jeho konkrétní nasazení pro ověřování pravosti.
- ***Key Encipherment*** – certifikát je určen k šifrování klíčů. Klasickým případem je elektronická obálka, kdy data jsou šifrována náhodným symetrickým šifrovacím klíčem, který je ke zprávě přibalen a zašifrován právě veřejným klíčem z takto označeného certifikátu.
- ***Data Encipherment*** – veřejný klíč z takto označeného certifikátu je určen pro šifrování dat (jiných než šifrovacích klíčů).
- ***Key Agreement*** – certifikát je určen pro výměnu klíčů (např. DH výměna klíčů).
- ***Key Certificate Sign*** – veřejný klíč uvedený v tomto certifikátu je určen pro verifikaci certifikátů. Tj. soukromý klíč příslušející k tomuto veřejnému klíči je možné použít pro podepisování certifikátů.
- ***CRL Sign*** – veřejný klíč uvedený v tomto certifikátu je určen k verifikaci CRL.
- ***Encipher Only*** – Tento bit se používá ve spojení s bitem *Key Agreement* (výměna klíčů). Výsledný dohodnutý symetrický klíč může být využit pouze k šifrování.
- ***Decipher Only*** – Tento bit se používá ve spojení s bitem *Key Agreemen*t (výměna klíčů). Výsledný dohodnutý symetrický klíč může být využit pouze k dešifrování.

Rozšíření se označí jako závažné. Tím se zamezí použití certifikátu k jiným účelům než k účelům vyznačeným v certifikátu.

2.5.4 Rozšířené použití klíče

Je obecnějším řešením pro určení účelů, k jakým je certifikát určen. Toto rozšíření může obsahovat sekvenci identifikátorů objektů specifikujících způsoby konkrétního použití veřejného klíče.

Toto rozšíření je předepsáno používat u některých dalších autorit. Např. autority pro vydávání časových razítek (TSA) vyžadují, aby certifikáty jejich serverů měly pomocí tohoto rozšíření explicitně vyjádřeno, že se mohou používat k tomuto účelu.

2.5.5 *Alternativní jméno předmětu*

Toto rozšíření umožňuje vložit do certifikátu další jedinečná jména držitele certifikátu. Např. jedinečné jméno ve světě e-mailové komunikace (e-mailovou adresu); jedinečné jméno v DNS světě důležitém zejména pro certifikáty počítačů (DNS jméno); další jedinečné jméno stejného tvaru jaký se používá pro předmět certifikátu (*Direktory*

name) atd. Důležité je, že alternativních jmen může být uvedeno i více.

Při vydávání certifikátu nesmí být opomenuta ani kontrola údajů uvedených v tomto rozšíření.

Měli bychom upustit od zlozvyku uvádět adresy elektronické pošty a DNS jména do předmětu certifikátu, ale uvádět je výhradně zde v rozšíření Alternativní jméno předmětu.

Může být zde uvedeno:

- ***Other Name*** – jiný identifikační údaj,
- ***rfc822 Name*** – adresa elektronické pošty dle RFC-822 (např. lidostalek@csob.cz),
- ***DNS Name*** – DNS jméno (např. jméno serveru www.firma.cz),
- ***X.400 Address*** – adresa elektronické pošty podle norem řady X.400,
- ***Directory Name*** – adresářové jméno podle norem řady X.500, tj. má stejný formát jako má předmět nebo vydavatel certifikátu,
- ***EDI Party Name*** – jméno podle norem EDI,
- ***Uniform Ressource Identifier*** – URI (např. http://www.firma.cz),
- ***IP Address***. Pro IPv4 obsahuje přesně 4 bajty IP-adresy verze 4, pro IPv6 obsahuje 16 bajtů IP-adresy verze 6, tj. jednotlivé bajty nejsou odděleny oddělovači ani převedeny do desítkové soustavy,
- ***Registered ID*** – identifikátor objektu.

2.5.6 Certifikační politiky (certifikační zásady)

Toto rozšíření obsahuje identifikátor dokumentu Certifikační politika. Navíc může obsahovat i hypertextový odkaz na tento dokument, který není součástí certifikátu. Tj. do výpočtu elektronického podpisu certifikátu je zahrnuto pouze toto rozšíření a nikoliv celý dokument. Není tedy zaručeno, že certifikační autorita tento dokument nezmění po vydání certifikátu.

Certifikační politika je dokument specifikující postupy, praktiky a cíle sloužící k ověření certifikátu před tím než je použit. Tj. pravidla, za kterých CA vydává certifikáty a zejména jak za vydané certifikáty ručí. Certifikační politika je dokument vydaný certifikační autoritou. Na rozdíl od některých jiných dokumentů CA je certifikační politika veřejným dokumentů zpravidla vystaveným na Internetu (na webu provozovatele certifikační autority).

Rozšíření Certifikační politiky v certifikátu:

- Není uvedeno. To je např. případ certifikačních autorit, které jsou součástí Microsoft server. Tyto certifikační autority nepředpokládají tvorbu pracné certifikační politiky při instalaci certifikační autority. Namísto certifikační politiky vyplňují do certifikátů své privátní rozšíření: *Ceritficate template*.
- V certifikátu je identifikátor certifikační politiky a případný hypertextový odkaz na tuto politiku, která je vystavena v Internetu. Propracovaná certifikační politika je totiž o několik řádů větší, než je velikost samotného certifikátu, takže její umístění do certifikátu by bylo neefektivní.
- V certifikátu je jen 200 znaků nepřesahující prohlášení vydavatele, které nahrazuje certifikační politiku (např.: „*Pouze pro testovací účely*“).

2.5.7 Mapování zásad

Toto rozšíření se používá pouze v certifikátech certifikačních autorit. Má význam v případě, že certifikát certifikační autority je podepsán jinou CA. V takovém případě je pravděpodobné, že nadřízená certifikační autorita si vydá jinou certifikační politiku než podřízená certifikační autorita. Při ověřování řetězce certifikátů je pak důležité, aby certifikační politiky jednotlivých certifikátů v řetězci byly konzistentní (si vzájemně odpovídaly).

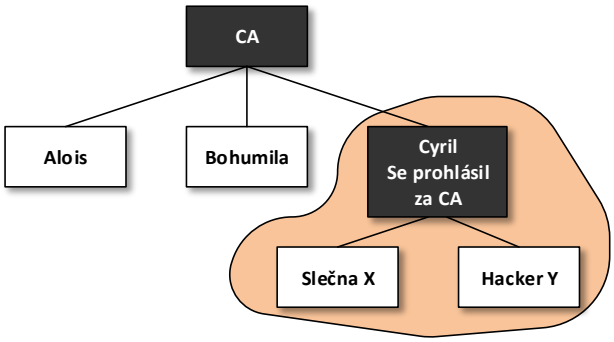
Jelikož každá CA má pro své certifikační politiky své identifikátory objektu, tak úkolem rozšíření Mapování zásad je sdělit, že ta a ta certifikační politika vydavatele (nadřízené CA – *issuerDomainPolicy*) je srovnatelná s tou a tou certifikační politikou předmětu (podřízené CA – *subjectDomainPolicy*).

2.5.8 Omezení využívání certifikátu (Constrains)

CA svým elektronickým podpisem ručí za údaje uvedené ve vydaných certifikátech. Na obrázku Obrázek 2.9 je znázorněno nebezpečí, když CA vydává certifikáty svým uživatelům: Aloisovi, Bohumile a Cyrilovi. Cyril v záchvatu žárlivosti se svévolně prohlásí za certifikační autoritu a vydá certifikáty dalším uživatelům: slečně X a hackerovi Y. Problém je v tom, že CA nepřímo ručí i za takto vydané certifikáty pro uživatele X a Y. Např. slečna X obdrží řetězec certifikátů obsahující: certifikát slečny X, certifikát Cyrila a certifikát CA. Certifikát slečny X se verifikuje pomocí certifikátu Cyrila. Certifikát Cyrila se následně verifikuje pomocí certifikátu CA. Čili není problém ověřit platnost takto vydaný certifikátu Cyrilem, ale problém je v zodpovědnosti CA za takto vydané certifikáty.

Jeden mechanismus, jak takovémuto počínání uživatele zamezit, jsme již popsali pomocí rozšíření Použití klíče. Tímto rozšířením v certifikátech vydávaných uživatelům omezíme použití jejich klíče tak, že jej není možné používat k verifikaci certifikátů.

Rozšíření mající v názvu slovo české slovo „omezení“ nebo anglické „*constrains*“ omezuje nejen



Obrázek 2.9 Cyril se svévolně prohlásil za certifikační autoritu

svévolným prohlášením uživatelů za certifikační autority, ale v případě, že i když cílevědomě vystavuje certifikát podřízené certifikační autoritě, tak omezíme její působnost výhradně na sjednané oblasti.

2.5.8.1 Základní omezení

Rozšíření Základní omezení (*Basic constrains*) umožňuje označit certifikát tak, aby bylo zřejmé, zdali se jedná o certifikát CA nebo koncového uživatele. V případě certifikátu CA umožňuje určit, kolik může mít tato certifikační autorita podřízených CA. Např. je-li tato položka nastavena na nulu, pak tato CA může vystavovat certifikáty jen koncovým uživatelům (nikoliv podřízeným certifikačním autoritám).

Toto rozšíření se využívá výhradně u certifikátů certifikačních autorit.

2.5.8.2 Omezení jmen

Certifikát certifikační autority se tímto omezením omezuje na vydávání certifikátů uživatelům splňujícím v jejich certifikátech zadané podmínky pro jedinečná jména či alternativní jména předmětu.

Omezovat lze např. na DNS jména patřící do domény .firma.cz. Lze rovněž omezovat jména poštovních schránek na poštovní adresy patřící určité doméně. Lze omezovat i IP-adresy apod. Toto rozšíření může být využito pouze v certifikátech CA.

2.5.9 Distribuční místa seznamu odvolaných certifikátů

Seznam odvolaných certifikátů (CRL) může vystavovat buď sama certifikační autorita nebo vystavováním CRL může pověřit jinou autoritu (např. autoritu pro vydávání CRL).

Toto rozšíření obsahuje seznam distribučních míst, na kterých je vystaven seznam odvolaných certifikátů (CRL). Distribuční místo je zpravidla reprezentováno URI.

2.5.10 *Subject directory attributes*

Jedná se o rozšíření obsahující další atributy (viz jedinečné jméno). Rozšíření obsahuje sekvenci těchto atributů. Norma RFC-3739 pro kvalifikované certifikáty však zavádí některé specifické atributy pro toto rozšíření.

Pomocí rozšíření S*ubject directory attributes* lze řešit tzv. problém uživatelských práv. Problém spočívá v tom, že certifikát slouží k prokázání totožnosti držitele certifikátu (na základě faktu, že uživatel má k dispozici odpovídající soukromý klíč). Prokázání totožnosti ještě nic neříká o tom, jestli mám nějaká přístupová práva ke konkrétní aplikaci. Podobně jako na základě občanského průkazu mohu prokázat svou totožnost, ale to ještě nic nevypovídá o tom, jestli jsem např. oprávněn vstupovat do nějaké budovy. V případě občanského průkazu může být takové přístupové právo vyznačeno v občanském průkazu. Jiným řešením je vydání průkazu ke vstupu. V takovém průkazu pak jsou opsány mé identifikační údaje z občanského průkazu a dále jsou vyznačeny, má přístupová práva. Já se mohu prokázat občanským průkazem a následně podle průkazu ke vstupu jsem buď vpuštěn, nebo ne.

V případě certifikátů máme opět obě možnosti. Právě v rozšíření „*Subject directory attributes*“ můžeme vyznačit např. přístupová práva přímo v certifikátu. Jinou možnosti je využít atributové certifikáty, které jsou pak obdobou dalšího průkazu.

2.5.11 Přístup k informacím úřadu (*Authority Information Access - AIA*)

Toto rozšíření může mít několik funkcí. V praxi se zpravidla nejčastěji využívají následující dvě možnosti:

- Pokud máme ověřit platnost certifikátu, pak potřebujeme mít k dispozici certifikát certifikační autority, která jej vydala. Ten můžeme mít např. již předem uložen

na našem počítači, ale pokud jej nemáme, pak jsme ve frapantní situaci. Odkud jej vzít? A právě v rozšíření AIA může být odkaz (URL) na kterém se potřebný certifikát certifikační autority nachází.
- Druhou možností je do tohoto rozšíření uvést odkaz na OCSP server, pomocí kterého můžeme OnLine zjišťovat, není-li náhodou certifikát odvolán.

2.5.12 Název šablony certifikátu

Certifikační autority dodávané firmou Microsoft vydávají certifikáty k různým účelům, např.: certifikáty CA, certifikáty podřízených CA, certifikát pro přihlášení uživatele do Windows, certifikát pouze pro podpis uživatele (*UserSignature*) atd. Obecně by asi bylo možné pro každý účel napsat certifikační politiku a tu vyznačit v certifikátu. Microsoft šel jinou cestou - vytvořil výčet všech účelů. Pro každý účel ze seznamu pak má konkrétní šablonu.

2.5.13 Biometrické informace

Jedná se o rozšíření pro kvalifikované certifikáty. V certifikátu nejsou uloženy samotné biometrické vlastnosti držitele certifikátu, ale pouze otisky z příslušných biometrických vlastností držitele certifikátu. U příslušného otisku může být uvedeno rovněž URI, na kterém je celý biometrický údaj kompletně k dispozici.

2.5.14 Qualified Certificate Statements

Toto rozšíření by mělo vyjadřovat, že se jedná o kvalifikovaný certifikát. Rozšíření obsahuje sekvenci příznaků kvalifikace certifikátu.

Každý příznak se skládá z identifikátoru objektu a parametrů definovaných při zavedení tohoto identifikátoru objektů. RFC-3739 zavádí pro identifikaci registrační autority. Další příznaky by měl zavést zákon (resp. související vyhlášky) země, ve které se kvalifikované certifikáty vydávají.

Označit certifikát jako kvalifikovaný je též alternativně možné pomocí rozšíření Certifikační politiky. V certifikační politice pak konkrétně uvedeme, že se jedná o kvalifikované certifikáty.

2.6 Kvalifikované certifikáty

Kvalifikovaný certifikát je zvláštní typ certifikátu, které používá Evropská unie ve své legislativě. Zvláštní není ani svou syntaxí (ta je ve své podstatě podmnožinou certifikátu popsaného v předchozí kapitole, tj. certifikátu dle RFC-5280). Zvláštnost je v právní oblasti. Cílem je po právní stránce nahradit rukou psaný podpis elektronickým podpisem, který se verifikuje právě kvalifikovaným certifikátem. Jádro myšlenek ohledně kvalifikovaných certifikátů je uvedeno v RFC-3739.

Kvalifikovaný certifikát obsahuje identifikaci držitele certifikátu založenou na oficiální identifikaci člověka nebo na jeho pseudonymu. Certifikační autorita vždy zná konkrétní osobu, které certifikát vydala.

Předmět certifikátu musí být jednoznačný pro konkrétní osobu, tj. dvě různé osoby nemohou mít vydán certifikát se shodným předmětem. Tato podmínka musí být splněna po celou dobu existence konkrétní CA. Pro docílení této podmínky je možné použít atribut *serialNumber* (nezaměňovat s položkou sériové číslo certifikátu). Kdyby dvě osoby měly mít stejné předměty, pak se odliší hodnotou v položce *serialNumber*.

U kvalifikovaných certifikátů nestačí, aby byl pouze předmět jednoznačný pro konkrétní osobu, ale certifikační autorita nesmí vydat dvěma různým osobám certifikát, který by měl stejný veřejný klíč. Tj. certifikační autorita musí po dobu své existence archivovat i veřejné klíče, které uživatelům podepsala. U veřejných klíčů musí mít i informaci, pro jaké algoritmy se budou používat, aby mohla porovnávat klíče, zdali již nejsou použité.

2.6.1 EV certifikáty

Princip PKI je skvělý, ale vždy závisí na implementaci. Vždy se budou vyskytovat důvěryhodné, méně důvěryhodné i nedůvěryhodné certifikační autority. Konsorcium výrobců internetových prohlížečů *CA/Browser Forum* se dohodlo na podmínkách, za kterých budou z jejich pohledu vydané certifikáty více důvěryhodné. Kromě požadavků na ověření předmětu certifikátu vydali i technická doporučení pro tzv. *Extended Validation Certificate (EV).* Pokud prohlížečem surfujete na serverech, které mají vydaný EV certifikát, tak pole s URL zezelená radostí.

EV certifikát by měl splňovat:

- Musí mít rozšíření certificatePolicies
- Může být: certificatePolicies:policyQualifiers:policyQualifierId
- Rozšíření basicConstraints nesmí mít položku cA fis hodnotou true.
- Rozšíření keyUsage nesmí obsahovat keyCertSign nebo cRLSign
- Rozšíření extKeyUsage musí být id-kp-serverAuth nebo/a id-kp-clientAuth (může být i id-kp-emailProtection) – nic víc!

- Musím mít rozšíření AIA
- Musí mít rozšíření CRL Distribution Points

2.6.2 eIDAS - certifikáty pro autentizaci internetových stránek

Nařízení EU eIDAS zase nařizuje, v případě kvalifikovaných certifikátů, pro certifikáty webových stránek používat certifikáty, které splňují následující podmínky.

- označení, že se certifikát vydává jako kvalifikovaný certifikát pro autentizaci internetových stránek
- Vydavatel (Název, IČO,..)
- Jméno/pseudonym/název právnické osoby
- Adresa
- DNS domény, které osoba provozuje
- Platnost
- Sériové číslo
- Zaručený podpis
- AIA (Authority Information Access, tj. URL s certifikátem vydávající CA)
- CRL, OCSP

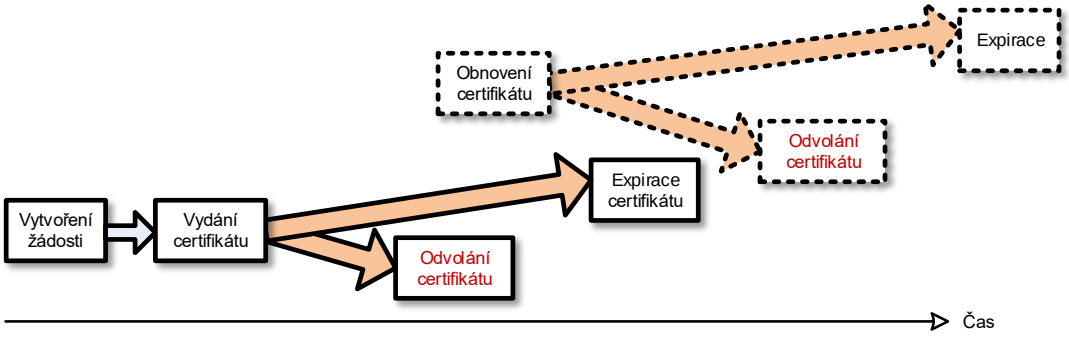
2.6.3 PSD2

Směrnice EU 2015/2366 o platebních službách na vnitřním trhu (PSD2) předepisuje formu certifikátu, na kterém spoléhá komunikace platebních institucí v EU. Tento certifikát musí splňovat:

- Musí být vydán dle eIDAS jako kvalifikovaný certifikát pro autentizaci webových stránek (resp. elektronické pečeti).
- Musí v anglickém jazyce obsahovat atributy:
    - s rolí poskytovatele (Poskytovatel iniciace platby, Poskytovatel informací o účtu atd.)
    - s názvem kompetentní autority, u které je poskytovatel platebních služeb registrován (např. v česku ČNB).

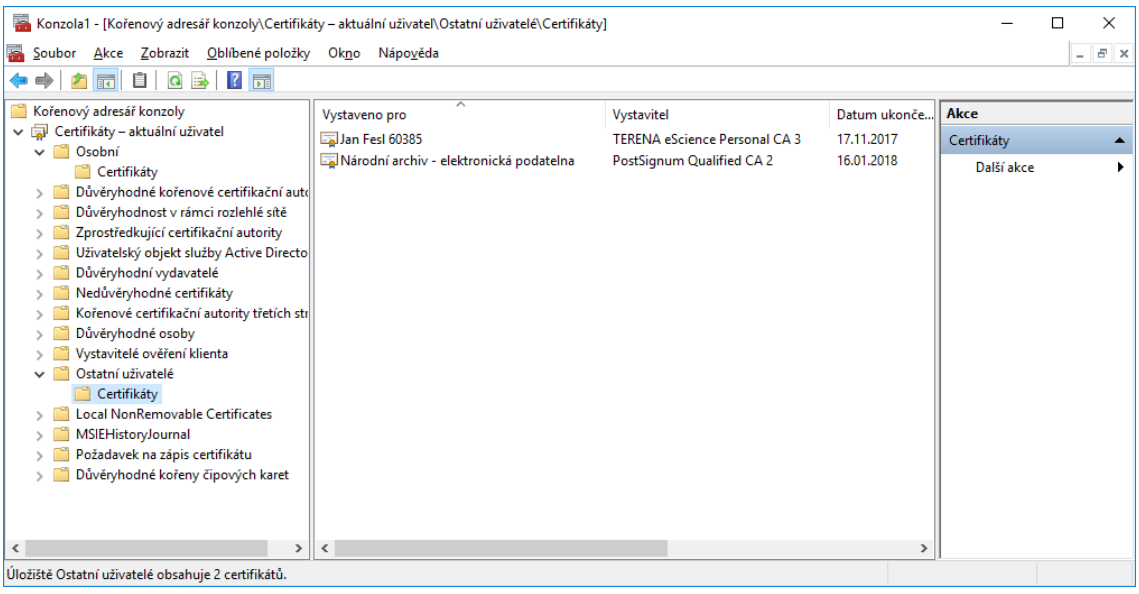
3 Životní cyklus certifikátu

Certifikát v průběhu času prochází několika fázemi tvořícími životní cyklus certifikátu (Obrázek 3.1). Životní cyklus certifikátu se skládá z následujících fází:



Obrázek 3.1 Životní cyklus certifikátu

1. **Vytvoření žádosti o certifikát.** Vytvoření žádosti může, ale i nemusí předcházet generování párových dat (je to sice málo běžné, ale párová data může generovat až certifikační autorita po obdržení žádosti o certifikát).
2. **Vydání certifikátu** a jeho případná publikace.
3. **Platnost certifikátu**. Poté co byl certifikát vydán, nemusí být ještě automaticky platný. Platnost certifikátu začíná v době uvedené v položce „od“ (*Not Before*) a končí buď vypršením platnosti certifikátu, nebo odvoláním certifikátu.
4. **Vypršení platnosti certifikátu** (expirace certifikátu) nastane po uplynutí doby „do“ (*Not After*) uvedené v certifikátu.
5. **Odvoláním certifikátu** před uplynutím jeho původně deklarované doby platnosti. Certifikát odvolává certifikační autorita zpravidla tím, že identifikaci certifikátu zveřejní na seznamu odvolaných certifikátů (CRL). Odvolaný certifikát se uvádí na všech CRL po dobu jeho původní platnosti Certifikační autorita odvolává certifikát:
   - Buď ze svého rozhodnutí. Např.:
     - Jiný uživatel požádal o certifikaci již certifikovaného veřejného klíče.
     - Certifikační autorita zjistila, že údaje v certifikátu nadále nejsou pravdivé (např. zaměstnanec rozvázal pracovní poměr a vlastní zaměstnanecký certifikát, tj. certifikát s uvedeným atributem „Organization“).
   - Nebo na žádost držitele certifikátu. Např.:
     - Uživatel si již nepřeje, aby certifikát dále platil z osobních důvodů.
     - Byl kompromitován soukromý klíč uživatele.



Obrázek 3.2 Výpis certifikátů ve Windows pomocí programu mmc

- Byl zničen soukromý klíč uživatele.

Existuje i možnost pozastavení platnosti certifikátu. Pozastavený certifikát se rovněž uvádí na všech CRL až do vypršení původní platnosti certifikátu. Avšak v době pozastavení platnosti se na CRL uvádí s příznakem pozastavení platnosti a v době po obnovení platnosti se na CRL uvádí s příznakem obnovení platnosti.

Pokud je certifikát ještě platný, pak jej můžeme využít k obnovení certifikátu. Co to znamená? Pokud žádáme o vydání certifikátu, pak zpravidla musíme prokázat svou totožnost, aby certifikační autorita mohla ověřit údaje v žádosti o certifikát. Pakliže máme platný certifikát, pak certifikační autorita již ověřila naší totožnost a tak stačí, když využijeme platný certifikát k opětovnému prokázání naší totožnosti. Pokud jsme se např. v případě vydání prvního certifikátu museli osobně dostavit k prokázání totožnosti, pak v případě obnovení certifikátu to už zpravidla není nutné – prokážeme se elektronicky za využití platného certifikátu.

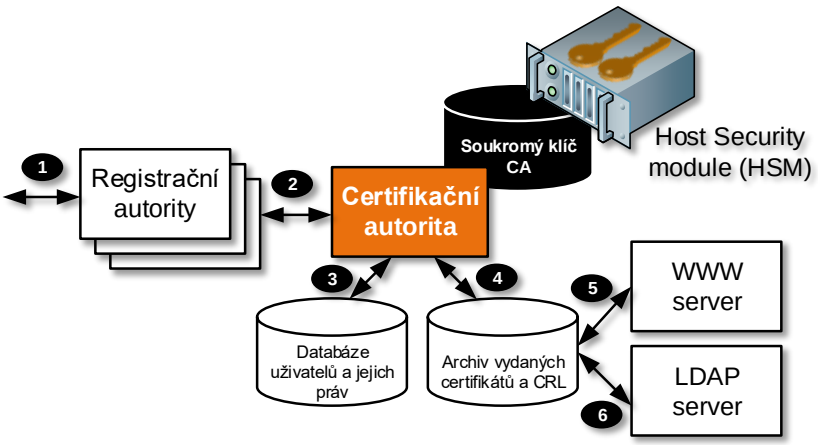
Certifikační autorita zpravidla uvede stejný předmět obnoveného certifikátu, jako měl původní certifikát. (Přesné podmínky jak certifikační autorita postupuje při vydávání a obnovování certifikátu ale najdeme v certifikační politice příslušné certifikační autority.)

Pokud používáme více certifikátů současně (např. šifrovací, podpisový, autentizační apod.), pak stačí, když je nám vydán např. podpisový certifikát a ostatní certifikáty si vydáme jako další certifikáty s tím, že totožnost již nemusíme prokazovat osobně, ale postačí ji prokázat elektronicky např. na bázi elektronického podpisu pomocí platného podpisového certifikátu.

3.1 Certifikát ve Windows

Ve Windows zpravidla bývají přípony souborů .cer, .crt, .der atp. asociovány s manažerem certifikátů. Po kliknutím na soubor s touto příponou se zobrazí obsah certifikátu (viz Obrázek 2.6).

Jiným řešením jak zobrazit certifikáty v systému Windows je spustit program mmc. Volbou „Přidat nebo odebrat modul snap-in“ přidáme modul „Certifikáty“ – viz obrázek Obrázek 3.2. Na obrázku jsou zobrazena úložiště certifikátů aktuálně přihlášeného uživatele. Kromě toho si správce může zobrazit i úložiště certifikátů místního počítače a úložiště služeb. Tato úložiště jsou důležitá zejména pro servery běžící na tomto počítači.



Obrázek 3.3 Struktura certifikační autority

Na obrázku vidíme jednotlivá úložiště certifikátů:

- **Osobní** – obsahuje osobní certifikáty aktuálně přihlášeného uživatele. Důležité je, že k osobním certifikátům zpravidla budeme mít i příslušné soukromé klíče.
- **Důvěryhodné kořenové certifikační autority** – obsahuje důvěryhodné kotvy z úložiště Kořenové certifikační autority třetích stran, důvěryhodné kotvy Microsoftu a podnikové důvěryhodné kotvy.
- **Důvěrnost v rámci rozlehlé sítě** – obsahuje CRL.
- **Zprostředkující certifikační autority** – obsahuje certifikáty mezilehlých certifikačních autorit, tj. certifikáty CA, které nejsou kořenové.
- **Uživatelský objekt služby Aktive Direktory** – obsahuje certifikáty, které má aktuální uživatel registrovány v ActiveDirectory.
- **Ostatní uživatelé –** obsahuje certifikáty ostatních uživatelů, které byly vydány důvěryhodnými CA. Toto úložiště může též využívat elektronická pošta pro vyhledávání certifikátů adresátů.
- **Požadavek na zápis certifikátu –** obsahuje žádosti o certifikáty.

V případě osobních certifikátů zpravidla budeme k certifikátu mít i soukromé klíče. Soukromé klíče se v operačních systémech Microsoft ukládají:

- Na disk. V tomto případě mohou být uloženy lokálně nebo jako součást uživatelského profile (ActiveDirectory).
- Na čipovou kartu, USB token či podobné zařízení.

V případě certifikátů počítače (serveru) se soukromé klíče ukládají do obdobných úložišť. Mohou být uloženy na disku serveru nebo v HSM modulu.

3.2 Certifikační a registrační autority

Certifikační autorita (CA) je nezávislá třetí strana, která vydává certifikáty. Slovní spojení „certifikační autorita“ lze ale chápat dvojím způsobem: buď jako aplikaci (vydávající certifikáty) nebo jako instituci (zajišťující proces vydávání certifikátů). Jako instituce může být CA realizována jako samostatná firma nebo jako samostatný útvar v rámci firmy.

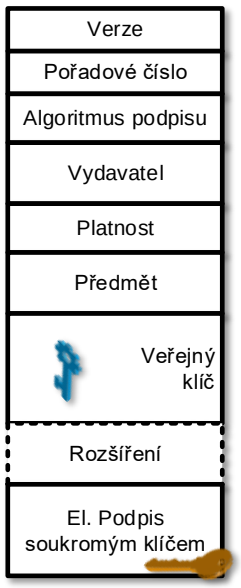
Certifikační autorita jako instituce se skládá z několika základních částí:

6. Registračních autorit (RA), které jsou často realizovány podobně jako bankovní přepážky. Na RA se dostavují žadatelé o certifikáty se svými žádostmi. RA mohou ověřovat totožnost žadatelů. RA následně zprostředkují vydání certifikátu (může být i vyžadováno aby se na RA dostavili osobně). Vydaný certifikát bývá skrze RA předán žadateli (ze kterého se tím stane držitel certifikátu). RA mohou být též organizovány i jako servery a uživatel s nimi komunikuje elektronicky.
7. Jádrem CA je certifikační autorita jako aplikace vydávající certifikáty, které jsou elektronicky podepisovány soukromým klíčem CA. Soukromý klíč CA je tak největším aktivem CA, které je nutné odpovídajícím způsobem chránit. Pro ochranu soukromého klíče CA se často využívají HSM.
8. CA udržuje databázi uživatelů a auditní záznamy o činnosti CA včetně případných účtovacích informací pro fakturaci poskytovaných služeb.
9. CA udržuje archiv vydaných certifikátů a CRL.
10. Archiv vydaných certifikátů a CRL může být dostupný skrze webové rozhraní.
11. Archiv vydaných certifikátů a CRL může být dostupný skrze protokol LDAP.

Kromě vydávání certifikátu certifikační autorita může též zajišťovat mechanismus odvolávání certifikátů.

4 Žádost o certifikát

Na obrázku (Obrázek 4.1) je schematicky znázorněna datová struktura certifikátu. Jednotlivé položky certifikátu musí certifikační autorita řádně naplnit před tím, než výsledek digitálně podepíše. O certifikát může žadatel žádat dvojím způsobem. Buď předloží datovou strukturu nazývanou žádost o certifikát, nebo nikoliv.



Obrázek 4.1 Struktura certifikátu

Zabývejme se nejprve druhým případem, kdy žadatel žádnou datovou strukturu nepředkládá a certifikační autorita vše zařídí za něj. Tento postup využívají některé firmy na svých intranetech pro vystavování certifikátů svým zaměstnancům.

Zajímavý je případ některých členských zemí EÚ, které takto vystavují dokonce občanské průkazy. Jak certifikační autority postupují, když žadatelé nepředkládají žádost o certifikát v elektronické podobě? Žadatel se dostaví na úřad, předloží své osobní údaje, tj. prokáže svou totožnost a žádá o vystavení nového občanského průkazu.

Během výroby občanského průkazu jsou za žadatele vygenerována párová data a je mu vystaven certifikát. Uživatel následně obdrží občanský průkaz ve formě čipové karty s párovými daty i vystavenými certifikáty. Zvenku karta nahrazuje stávající tištěné občanské průkazy a uvnitř je čip se soukromými klíči a vydanými certifikáty. Jedná se v podstatě o obdobu vydávání platebních karet. Tento způsob vyžaduje vysoké nároky na bezpečnost zařízení, na kterých se generují párová data i maximální bezpečnost všech procesů manipulujících s kartou než se karta dostane do ruky svému oprávněnému držiteli.

Dále se budeme zabývat již jen případem, kdy žadatel předkládá elektronickou žádost o certifikát. Žádost o certifikát pak obsahuje informace (nebo jejich části), které by si žadatel přál mít uvedeny v certifikátu. Je pak na certifikační politice CA, jestli obsah těchto položek zkopíruje (nebo nezkopíruje) do certifikátu. Nakonec některé informace do certifikátu doplní certifikační autorita ze své iniciativy a vydá kýžený certifikát.

Máme několik možností elektronický žádosti o certifikát: PEM, PKCS#10, CRMF, kořenový certifikát (paragraf 5.3) a SPK.

4.1 Údaje v žádosti o certifikát

Žádost o certifikát by mněla obsahovat:

- **Identifikační údaje žadatele**. Ty ve výsledném certifikátu budou uvedeny v předmětu certifikátu nebo v rozšíření Alternativním jméno předmětu.
- **Veřejný klíč** včetně příslušné identifikace asymetrického algoritmu, ke kterému je veřejný klíč určen.
- **Důkaz o držení příslušného soukromého klíče**.
- **Další údaje**, které si přeje uživatel do certifikátu vložit (např. použití klíče, rozšířené použití klíče apod.).
- Může obsahovat **důkaz o generování párových dat** bezpečným zařízením (např. čipovou kartou). Pro tento důkaz nemívají žádosti o certifikát separátní položku. Vždy se však najde nějaké rozšíření, do kterého je jej možné vložit.
- Pokud je vydání certifikátu zpoplatňováno, pak je třeba též předat **údaje potřebné pro fakturaci** (fakturační adresa, IČO, DIČ, číslo bankovního účtu pro inkaso apod.).
- **Hesla pro komunikaci s certifikační autoritou**. Jedná se zejména o:
    - Jednorázové heslo pro vystavení certifikátu, které je užitečné zejména v případě, že vydavatel certifikátů nechce uživatele obtěžovat častou návštěvou registračních autorit a chce využít strategii „jedna návštěva stačí“. Uživatel se totiž zpravidla poprvé bez problémů dostaví na registrační autoritu bez žádosti o certifikát pro informace. Avšak podruhé (s žádostí o certifikát) se mu tam už osobně nechce. Takže již při první návštěvě s ním registrační autorita sepíše veškeré podklady a místo vydání certifikátu mu vydá jednorázové heslo pro vydání certifikátu. Uživatel si pak z tepla domova či kanceláře odešle žádost přes Internet a doplní jí jednorázovým heslem o vydání certifikátu. CA následně zkontroluje, zdali se shodují údaje, které vyplnil při první návštěvě, s údaji v žádosti. A navíc zkontro-

luje jednorázové heslo. Když je vše v pořádku, pak vydá certifikát, který opět uživateli zašle přes Internet.

- Jednorázové heslo pro odvolání certifikátu je velice užitečné v okamžiku, kdy je uživateli zcizen soukromý klíč. Např. mu byla odcizena PKI čipová karta na níž měl fixem napsán PIN. V takovém případě je rychlé využití jednorázového hesla pro zneplatnění certifikátu přímo k nezaplacení. Uživatel se pak může na CA obrátit libovolným komunikačním kanálem (telefonicky, faxem, mailem, webem apod.) a požádat o odvolání svého certifikátu. Autentizuje se přitom jednorázovým heslem pro zneplatnění certifikátu.
- Stálé heslo pro osobní (neelektronickou) komunikaci uživatele s CA. Na základě autentizace tímto heslem může CA poskytovat různou podporu svým uživatelům.
- Kromě stálého hesla si uživatelé často volí ještě tzv. frázi. Fráze je užitečná v okamžiku, kdy uživatel přijde úplně o vše včetně jednorázových a stálých hesel. Jako fráze se volí např. dívčí jméno tchýně apod. Pokud i to zapomene, pak se má zpravidla koho zeptat, postupuje-li obezřetně.

4.2 Důkaz o vlastnictví soukromého klíče

Prvním dotazem je proč by měl žadatel o certifikát předkládat důkaz o držení příslušného soukromého klíče? Na první pohled se může zdát nesmyslné, proč by někdo žádal o vystavení certifikátu veřejného klíče, když k němu nemá odpovídající soukromý klíč!

Představte si, že by žadatel o certifikát vygeneroval párová data shodná, jako vygeneroval jiný žadatel, kterému již byl vystaven certifikát veřejného klíče. Pokud by certifikační autorita vydala certifikáty se stejným veřejným klíčem dvěma různým osobám, pak by to bylo velice frapantní. Oba by měli stejný soukromý klíč a mohli by jeden za druhého např. podepisovat dokumenty.

Případy, kdy si dva uživatelé vygenerují stejná párová data jsou téměř nemožné. Avšak to platí jen za předpokladu, že uživatelé používají kvalitní generátory náhodných čísel. Vinou nekvalitního software se nemožné může stát realitou. Certifikační autority zpravidla kontrolují, jestli předkládaný veřejný klíč již necertifikovaly (musí být shodný i algoritmus). Pokud ano a certifikát je platný, pak rozumná certifikační autorita původní certifikát okamžitě odvolá a novou žádost zamítne.

Takže pokud by CA nekontrolovala držení příslušného soukromého klíče, pak by se útočník dostavil s žádostí o certifikát s veřejným klíčem uživatele, kterému chce uškodit a CA by příslušný certifikát ke škodě uživatele odvolala.

Nejčastějším typem důkazu o držení příslušného soukromého klíče je elektronický podpis ze struktury obsahující veřejný klíč. V anglické terminologii se žádosti o certifikát obsahující důkaz o držení příslušného soukromého klíče na bázi elektronického podpisu označují jako **CSR** – *Certificate Signing Request*.

Důkaz o vlastnictví soukromého klíče je třeba vytvářet na jiném principu pro klíče určené k elektronickému podepisování, pro klíče určené k šifrování a pro klíče určené pro výměnu klíčů (např. Diffie-Hellmanův algoritmus).

4.2.1 Verifikaci důkazu provedla RA jinou cestou

Je také možné, že důkaz o vlastnictví soukromého klíče provede registrační autorita jinou cestou. Pak žádost o certifikát nemusí obsahovat žádný důkaz. Praktické ale je, aby skutečnost, že RA ověřila důkaz vlastnictví soukromého klíče, byla v žádosti stvrzena.

4.2.2 Důkaz založený na elektronickém podpisu

V případě, že žádost obsahuje veřejný klíč určený pro ověření elektronického podpisu, vytvoří se jako důkaz elektronický podpis z žádosti. Problém je jen u žádostí, které neobsahují veřejný klíč, pak se jako důkaz musí použít elektronický podpis z nějakého jiného řetězce.

4.2.3 Důkaz pro šifrovací klíče

V případě, že žádost obsahuje šifrovací klíč, je nutné provést důkaz na základě šifrování. V tomto případě máme následující možnosti:

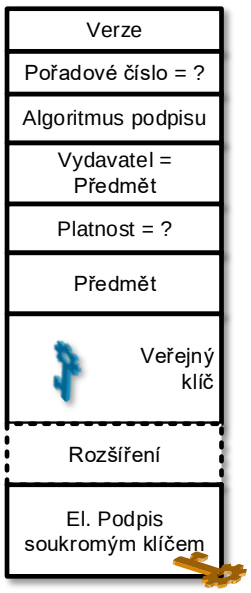
- Žadatel může poskytnout celou dvojici veřejný/soukromý klíč. Jelikož se jedná o šifrovací klíče, tak může využívat i možnost archivace soukromého klíče na CA.
- Žadatel využije dešifrování jako důkaz, že vlastní příslušný soukromý klíč. Jsou zde dvě metody:
  - Přímá metoda: CA vygeneruje náhodný dotaz, který šifruje veřejným klíčem. Žadatel jej dešifruje svým soukromým klí-

čem a nešifrovaný výsledek vrátí CA. Pokud CA obdrží řetězec, který vygenerovala, pak nikdo jiný než držitel soukromého klíče jej nemohl dešifrovat.

- Nepřímá metoda: CA vydá certifikát, který šifruje veřejným klíčem uvedeným v certifikátu a výsledek předá žadateli. Jedině žadatel mající k dispozici příslušný soukromý klíč je schopen takto šifrovaný certifikát dešifrovat a publikovat.

4.2.4 Důkaz na základě výměny klíčů

Pro výměnu klíčů (např. Diffie-Hellmanův algoritmus) je možné využít jak přímou, tak i nepřímou metodu uvedenou v předchozím odstavci. Obě strany (žadatel i CA/RA) si nejprve na základě výměny svých klíčů ustaví společný tajný šifrovací klíč. Tento klíč je pak možné využít pro symetrickou šifru, kterou je šifrován náhodný dotaz (přímá metoda) nebo vystavený certifikát (nepřímá metoda).



Obrázek 4.2 Kořenový certifikát

V případě algoritmů pro výměnu klíčů je možná ještě jedna alternativa. Výměnou klíčů si obě strany ustaví společné sdílené tajemství, které následně žadatel o certifikát použije pro výpočet kryptografického kontrolního součtu (MAC). CA/RA pak verifikuje tento kryptografický kontrolní součet.

Všechny metody vyžadovaly, aby obě strany měly vygenerovány svá párová data pro výměnu klíčů.

4.3 Kořenový certifikát

Kořenový[*)] (*selfsigned*) certifikát je certifikát, který si vydává sám žadatel o certifikát. Kořenový certifikát má stejnou datovou strukturu jako certifikát. Takže máme kořenové certifikáty X.509 verze 1, či X.509 verze 3.

Jelikož kořenové certifikáty si vydává sám uživatel, tak se tyto certifikáty poznají podle toho, že mají položku Předmět identickou s položkou Vydavatel.

Pokud chceme vytvořit žádost o certifikát ve tvaru kořenového certifikátu, pak zjistíme, že to není až tak jednoduché. Certifikát totiž musí mít některá pole, do kterých nebudeme vědět co vyplnit. Např. sériové číslo certifikátu, platnost certifikátu apod.

Jako důkaz o držení soukromého klíče uživatelem se v kořenovém certifikátu použije elektronický podpis certifikátu vytvořený soukromým klíčem příslušejícím k veřejnému klíči uvedeném v certifikátu. Verifikace tohoto podpisu se tedy provádí veřejným klíčem uvedeným v samotném kořenovém certifikátu.

Kořenový certifikát se v praxi nepoužívá jako žádost o certifikát, se kterou by žadatelé přicházeli na veřejnou CA žádat o vystavení certifikátu. Software jej však často používá vnitřně. Pokud žadatel vygeneruje žádost o certifikát, aby ji uplatnil na veřejné CA, pak po dobu mezi vygenerováním párových dat uživatelem a okamžikem, kdy CA vystaví certifikát, musí být veřejný klíč udržován na počítači žadatele. V tomto mezidobí bývá veřejný klíč často udržován ve formátu kořenového certifikátu. Kořenový certifikát bývá pak přepsán vydaným certifikátem (se stejným veřejným klíčem).

4.3.1 Pozor! Není kořenový certifikát jako kořenový certifikát

V předchozím odstavci jsme uvedli, že kořenový certifikát si vydává uživatel jako zvláštní typ žádosti o certifikát.

Termín kořenový certifikát v případě certifikátů CA má zcela jiný význam. Kořenové certifikáty jsou též certifikáty kořenových CA - jsou tou nejvyšší autoritou (tzv. důvěryhodnou kotvou). Tyto certifikáty mají rovněž shodné položky Vydavatel a Předmět, ale mají řádně vyplněny i všechny ostatní položky (pořadové číslo, platnost,…).

Formát obou typů kořenových certifikátů je týž, ale bezpečnostní význam je naprosto jiný!

4.4 PEM

PEM (*Privacy Enhancement for Internet Electronic Mail*) byl z dnešního pohledu prvním systémem PKI na Internetu, který pracoval s certifikáty X.509 (tehdy verze 1). Jeho počátky jsou v RFC-989 z února 1987. Postupně byl přepracováván až do RFC-1421 až RFC-1424.

[*)] Překlad slova *selfsigned* jako kořenový v tomto případě trochu kulhá, ale nevymysleli jsme lepší.

Protokol PEM jako žádost o certifikát využíval X.509 kořenový certifikát verze 1. Jakými hodnotami vyplnit sporná pole kořenového certifikátu předepisuje RFC-1424, které je pro tento případ inspirací dodnes.

Jenže certifikát X.509 je binární struktura a protokol PEM byl orientován na komunikaci elektronickou poštou, která je principiálně sedmibitová. Takže nejenom žádost o certifikát, ale všechna další binární data protokol PEM kódoval algoritmem Base64 (viz kap. 14) do sedmibitového tvaru. Výsledek byl uvozen a ukončen vždy řádkem, který mezi skupinou pomlček měl napsáno, o jakou strukturu se jedná:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----

<text>
-----END PRIVACY-ENHANCED MESSAGE-----
```

Protokol PEM se v praxi neujal (v té době byl masově používán systém PGP). Zbylo nám po něm ale označení, že nějaká struktura je ve formátu PEM. Míní se tím, že původní binární struktura byla kódována Base64 do sedmibitového tvaru. Takže v diskusi můžeme slyšet „pošli mi tu PKCS#10 žádost (či ten certifikát, to CRL,…) v PEM formátu“. A není tím míněno, že by žádost měla být ve tvaru kořenového certifikátu verze 1, ale pouze to, že binární data mají být kódována Base64.

4.5 PKCS#10

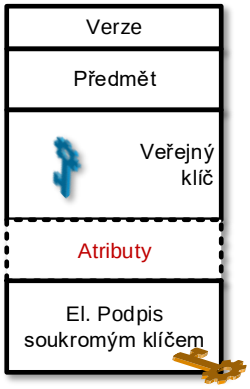
PKCS#10 (Obrázek 4.3) je konečně již opravdovou žádostí o certifikát. Velice zjednodušeně řečeno PKCS#10 je obdobou kořenového certifikátu s tím rozdílem, že sporná pole, do kterých jsme nevěděli co vyplnit, byla vypuštěna.

Zbyla nám tak následující pole:

- Verze, do které vyplňujeme verzi struktury PKCS#10. Tč. se má tato položka hodnotu 0.
- Předmět
- Veřejný klíč
- Atributy - tato položka obsahuje jednotlivé atributy. Jedním z atributů jsou rozšíření, která si žadatel přeje uvést do certifikátu.

Do položky Atributy můžeme navrhnout rozšíření, která si přejeme mít ve výsledném certifikátu. Pochopitelně závisí na certifikační politice CA, jak CA se zde uvedenými atributy naloží. Dalším typem atributu je jednorázového heslo pro odvolání certifikátu. Tento atribut se z pochopitelných důvodů ve vydaném certifikátu neobjeví.

Žádost PKCS#10 je podepsána soukromým klíčem, který přísluší k veřejnému klíči uvedenému v žádosti. To je obdobně jako v případě kořenových certifikátů důkaz, že žadatel vlastní příslušný soukromý klíč.



Obrázek 4.3 Žádost o certifikát formátu PKCS#10

PKCS#10 je podniková norma firmy RSA. Vyšla tedy ze světa algoritmu RSA. Algoritmus RSA umožňuje jak šifrování, tak i elektronický podpis. To je ale také základním omezením i toho jinak velice populárního formátu žádosti o certifikát. Je totiž obtížné tento formát použít pro žádost o certifikaci veřejných klíčů algoritmů, které neumožňují elektronický podpis. Proč se tento formát nehodí pro žádosti o certifikaci veřejných klíčů (např. i RSA), které nebudou určeny k ověřování elektronického podpisu? Asi si říkáte, proč takové komplikace s šifrovacími certifikáty. Ať to prostě uživatel podepíše a hotovo, a pak ať si ten certifikát používá jen pro šifrování. Jenže pokud žádáme o certifikací Diffie-Hellmanova veřejného čísla, tak ze soukromého DH čísla elektronický podpis prostě neuděláme, ať se budeme snažit sebevíc. A asymetrických algoritmů nepodporujících elektronický podpis je více.

Dále se PKCS#10 žádost nehodí pro případy, kdy jsou párová data generována až CA, tj. kdy uživatel v okamžiku žádosti párová data nemá k dispozici.

Obtížné je i obnovování certifikátů žádostí PKCS#10. Při obnovování certifikátů musíme předvést důkaz o držení nejenom nového soukromého klíče, ale i starého klíče (čímž prokazujeme svou totožnost – lépe řečeno kontinuitu totožnosti). To lze ale vyřešit vložením žádosti PKCS#10 např. do zprávy CMS, která je podepsána starým klíčem.

Norma PKCS#10 byla převzata nejprve jako RFC-2314 a později jako RFC-2986.

4.6 CRMF

CRMF (*Certificate Request Message Format - RFC 2511, RFC 4211*) je podstatně bohatší žádost než žádost PKCS#10. Řeší totiž některé již zmiňované

problémy, s nimiž se žádost PKCS#10 nedokáže vyrovnat. Žádost tvaru CRMF umožňuje jako součást žádosti zaslat dokonce i informace pro fakturaci.

4.7 CMC

Protokol CMC kompletně řeší dialog pro vystavování certifikátu. Pomocí tohoto protokolu je tedy rovněž možné vystavovat certifikáty.

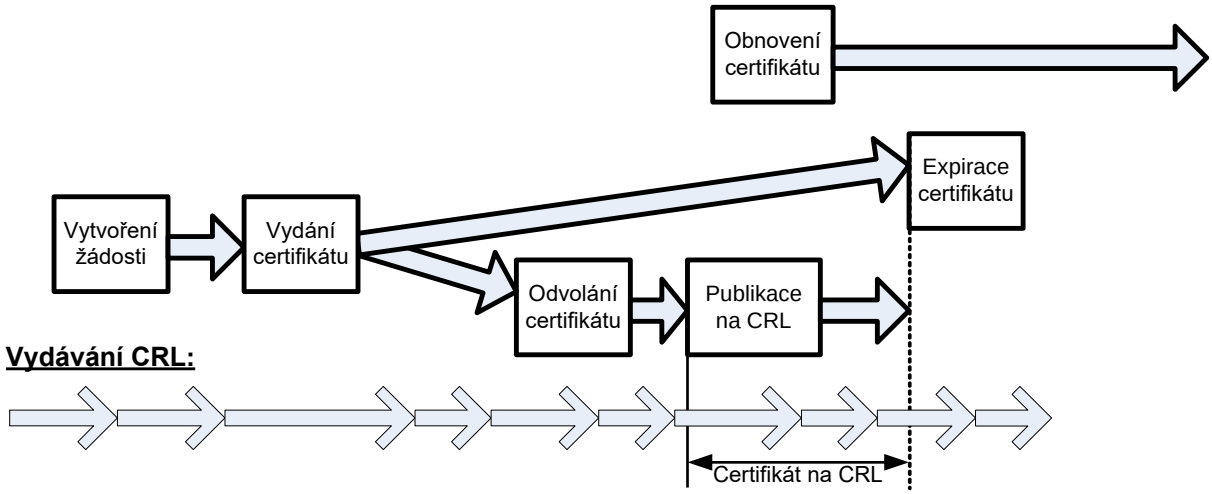
Tento protokol je v některých případech využíván i Microsoft CA (MSCA). A to zejména v případě, kdy je požadováno, aby žádost o certifikát obsahovala více podpisů: jeden pomocí klíče žadatele a druhý pomocí klíče RA (např. v případě tzv. *Smartcard Enrolment Station*).

5 Odvolávání certifikátu

Certifikát může ztratit svou platnost tak, že vyprší jeho původně deklarovaná doba platnosti, tj. uplyne čas `notAfter` uvedený v certifikátu. Před tím, než vyprší původně deklarovaná doba platnosti certifikátu, může být certifikát odvolán nebo může být jeho platnost pozastavena. Pozastavení certifikátů je však jen zřídka používaným institutem.

S odvoláváním certifikátu je to podobné jako s odvoláváním platební karty. Nejprve nám platební kartu někdo odcizí, aniž bychom si toho všimli. Pak si toho všimneme a nahlásíme ztrátu karty jejímu

- Nepřímá CRL vydává jiná než certifikační autorita, která vydala odvolávané certifikáty. Taková autorita se nazývá autoritou pro vydávání CRL.
- Úplné CRL obsahuje identifikace všech odvolaných certifikátů, jejichž původní doba platnosti dosud nevypršela.
- Přírůstkové CRL obsahuje identifikace jen těch odvolaných certifikátů, které byly odvolány od posledního úplného CRL
- Částečné (omezené) CRL obsahuje podmnožinu úplného CRL. Částečné CRL obsahuje rovněž kritérium, podle kterého byly vybrány identifikace certifikátů do částečného CRL.



Obrázek 5.1 Životní cyklus certifikátu a vydávání CRL

vydavateli. Vydavatel tuto informaci zpracuje a výsledkem je, že se karta dostane na stop list.

Každý obchodník je povinen si ověřit před tím, než přijme platbu prostřednictvím karty, zdali karta není na stop listu (automaticky to provádí platební terminál). Pokud si to neověří a karta na stop listu je, pak jdou náklady na jeho vrub. Veškeré diskuse jsou jen o tom, kdo ručí za kartu mezi jejím odcizením a okamžikem, kdy se dostane na stop list. Toto mezidobí může vzít na sebe vydavatel karty jako službu držiteli karty, karta se může pojistit apod.

Odvolané certifikáty jsou publikovány na seznamu odvolaných certifikátů (CRL) po celou dobu jejich původně deklarované platnosti. To jestli certifikační autorita vůbec umožňuje certifikáty odvolávat či s jakou periodou vydává CRL deklaruje certifikační autorita v dokumentu „Certifikační politika“.

Rozlišujeme několik druhů CRL:

- Přímá CRL vydává certifikační autorita, která vydala i odvolávané certifikáty.

V drtivé většině případů se budeme setkávat s přímými, úplnými CRL. Jestliže nebude uvedeno jinak, pak pokud budeme uvádět termín CRL, budeme mít na mysli přímé, úplné CRL.

Žádost o odvolání certifikátu nemusí být prováděna elektronickou formou, ale CA může vyžadovat např. osobní účast uživatele. Pokud se provádí elektronickou cestou, musí být žádost elektronicky podepsána soukromým klíčem odvolávaného certifikátu (v případě, že by takovou žádost provedl hacker neoprávněně, tj. uměl ji elektronicky podepsat, čili znal soukromý klíč, pak je pro uživatele jen dobře, že ji podal). Případně může být použito jednorázové heslo pro odvolání certifikátu.

Na obrázku Obrázek 5.2 je znázorněn proces odvolání certifikátu. Nejprve byl kompromitován privátní klíč. Tento okamžik se označuje jako okamžik Zcizení (kompromitace) soukromého klíče (*Invalidity date*). Poté si toho uživatel všiml a nahlásil to na CA. Tento okamžik se označuje jako Obdržení požadavku na odvolání (*Revocation*

date). Jestliže CA žádost o odvolání shledala za oprávněnou, pak identifikaci certifikátu zařadila na nejbližší vydávané CRL (Vydání CRL - *This update of CRL*).

I když si postižený uživatel kompromitace svého soukromého klíče nepovšiml třeba i řadu dní, od CA však očekává okamžitou publikaci svého certifikátu na CRL. Certifikační autority většinou volí jednu ze dvou strategii vydávání CRL:

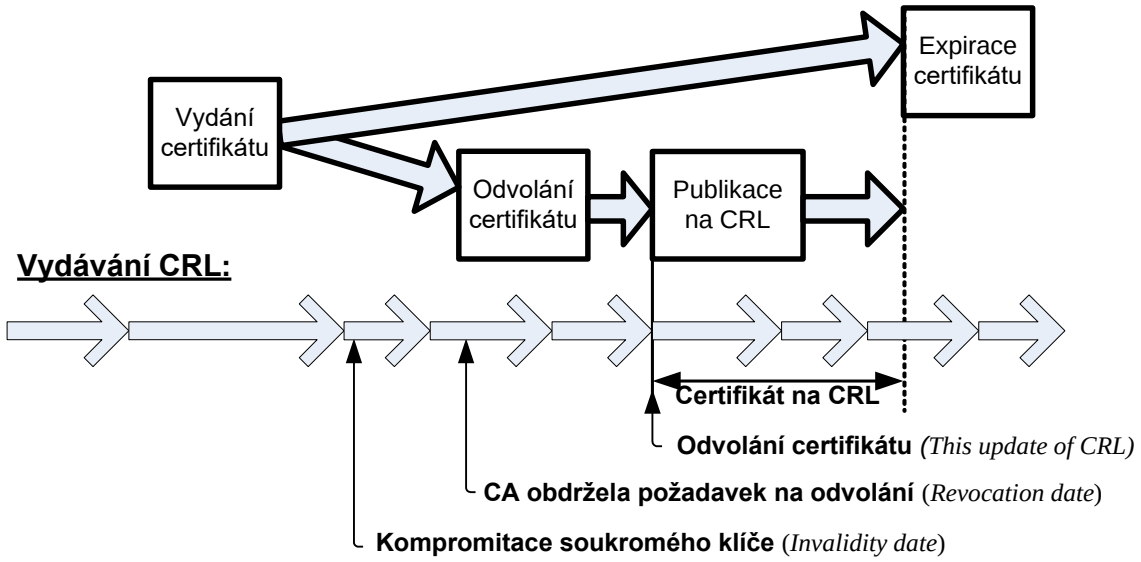
- CRL vydávají v podstatě v pravidelných intervalech a tak uživatel s kompromitovaným klíčem musí přetrpět interval do vydání dalšího CRL.
- CRL vydávají vždy po odvolání nějakého certifikátu. To však vyžaduje, aby ostatní uživatelé si skutečně před použitím každého certifikátu stáhli a zpracovali CRL, což je zase zdržuje při práci, protože stažení CRL vždy nějakou dobu trvá.

certifikát byl odvolán, když ještě není uveřejněn na CRL. Na tuto otázku neexistuje jednoznačná odpověď, ale většina lidí na to má vyhraněný názor.

5.1 Žádost o odvolání certifikátu

Při odvolávání certifikátu nejde o to, postupovat podle nějakých technických norem, ale jde především o rychlost, proto certifikační politiky jednotlivých CA bývají v této oblasti poměrně pružné. Neexistuje nějaká standardizovaná struktura s žádostí o odvolání certifikátu. Je několik možných způsobů odvolání certifikátu, jejichž použití většinou závisí na okolnostech, za kterých se certifikát odvolává:

- Pokud byl kompromitován soukromý klíč, ale máte jej k dispozici, pošlete žádost o odvolání certifikátu na CA např. elektronickou poštou. Zprávu elektronické pošty podepíšete soukromým klíčem příslušejícím odvolávanému certifikátu. Pokud se divíte, že něco podepisujete kompromitovaným klíčem, tak si uvědomte, že pokud by o odvolání požádal útočník, pak mu jen poděkujete. Omezení této metody:
  - Soukromý klíč musí být k dispozici.
  - Tuto metodu nelze použít, jestliže odvolávaný certifikát není určen k elektronickému podpisu. Pokud však máte vydán jiný podpisový certifikát, který je platný, pak může CA požádat o revokaci digitálně podepsanou zprávou za využití jiného platného podpisového certifikátu. Tuto metodu využívají zejména firmy pro odvolávání certifikátů



Obrázek 5.2 Jízdní řád odvolávání certifikátu

Realita však není tak romantická. Nejčastějším důvodem pro odvolání certifikátu kupodivu není kompromitace soukromého klíče ale ukončení pracovního poměru zaměstnance, který měl vystaven zaměstnanecký certifikát. Zaměstnavatelé jsou schopni takový certifikát odvolat v dostatečném předstihu.

Vydávání CRL je ve své podstatě dávkovou záležitostí. Vznikají tak různé protokoly pro On Line zjišťování stavu certifikátů. Základní otázkou však zůstává, může-li certifikační autorita jiným kanálem (např. On Line kanálem) poskytnout informaci, že

svých zaměstnanců – viz následující paragraf..

- Pokud se odvolává certifikát zaměstnance rozvazujícího pracovní poměr, pak jej pravděpodobně odvolává pověřený pracovník zaměstnavatele. K takovému odvolání stačí elektronicky podepsaná zpráva. Omezení této metody:
  - Pověřený pracovník zaměstnavatele musí mít vystaven odpovídající certifikát pro elektronický podpis.
  - Zaměstnavatel by měl mít uzavřenu smlouvu s CA o tom, kteří jeho zaměstnanci mohou žádat o odvolání certifikátů ostatních zaměstnanců.
- Některé CA s vydáním certifikátu též vystaví jednorázové heslo pro odvolání certifikátu. Jestliže toto heslo znáte, lze odvolat certifikát telefonicky, faxem pomocí webového formuláře nebo nepodepsanou elektronickou poštou s uvedením zmíněného jednorázového hesla. Omezení této metody:
  - S vydáním certifikátu musí být vydáno i příslušné jednorázové heslo pro odvolání certifikátu (či obecně heslo pro ne-elektronickou komunikaci uživatele s CA).
  - Uživatel toto heslo nesmí zapomenout.
- Neznáte-li ani jednorázové heslo, nezbývá, než se s doklady totožnosti vydat na registrační autoritu. Pokud přijdete úplně o vše, máte problém. V takovém případě to ale asi nebude ten největší problém, který máte.

Nesmíme zapomenout na případ, že CA odvolává certifikát ze své iniciativy. Takovým případem může být situace, kdy se objeví žádost o vydání certifikátu s již certifikovaným veřejným klíčem. Dalším případem je skutečnost, kdy CA zjistí, že údaje uvedené v certifikátu (např. v předmětu certifikátu) již nejsou platné. Takovým případem je i skutečnost, když o revokaci certifikátu požádá odcházející zaměstnanec. Pověřený pracovník zaměstnavatele vlastně neodvolává přímo certifikát, ale jen informuje CA o tom, že údaje v uvedeném certifikátu již nadále nebudou platné.

5.2 CRL

CRL je ve své podstatě něco jako úřední deska CA. CA na ni zveřejňuje odvolané certifikáty. Je-li jednou certifikát odvolán (zveřejněn na CRL), pak by měl být zveřejňován i na všech následujících CRL dokud by nevypršela jeho původní platnost.

Jenže je zde jeden drobný zádrhel. Některé CA umožňují platnost certifikátu pozastavit. To celý proces nesmírně komplikuje. Pokud totiž certifikát opět prohlásíte za platný, pak resuscitovaný certifikát (*Removed from CRL*) musí být rovněž uváděn na CRL až do vypršení jeho původní platnosti. Ponaučení je takové, že pokud je to jen možné, nevytváříme certifikační politiky umožňující pozastavení platnosti certifikátu. Snažíme se držet strategii, je-li jednou certifikát odvolán, pak už to nelze vzít zpět.

Bohužel to vždy nejde. Takovou zvláštní situací, která lze řešit pozastavením platnosti certifikátu je případ CA, která vydává certifikáty bez elektronický žádosti o certifikáty. Certifikát je vydán na čipové kartě a platnost by měl obdržet až po převzetí párových dat uživatelem. Jenže nikdo předem neví, kdy si občan pro své doklady dojde na úřad.

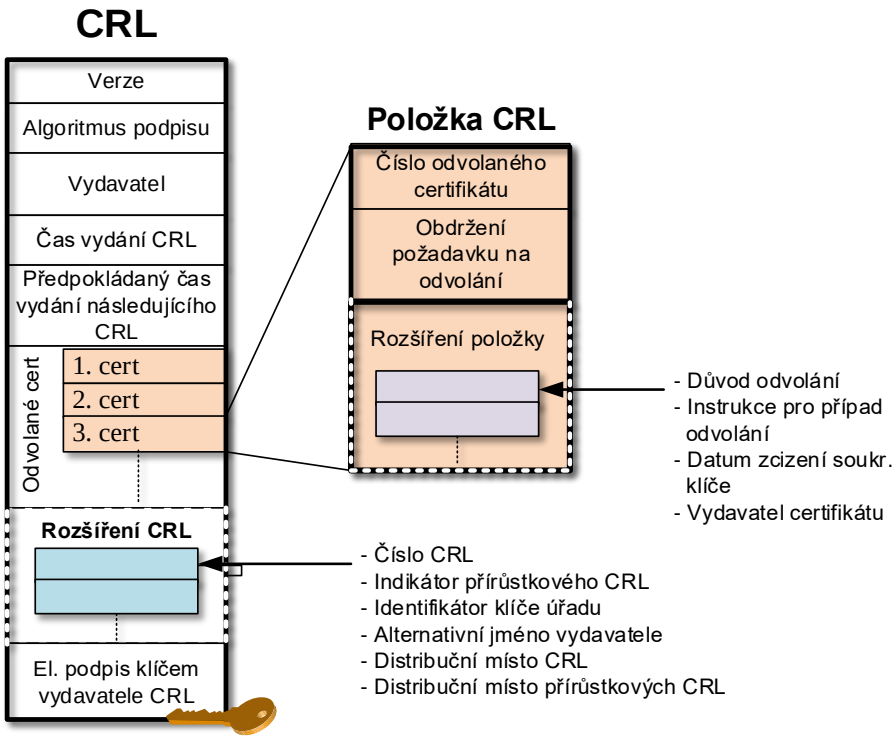
CRL zpravidla vydávají certifikační autority. Certifikační autority však nemusí vydávat CRL samy, ale mohou touto službou pověřit jinou autoritu – autoritu pro vydávání CRL. Ta pak vydává tzv. nepřímá CRL. U nepřímých CRL se zpravidla postupuje tak, že CA vydá certifikát serveru na vydávání svých CRL. Tento certifikát má v rozšíření Použití klíče nastaven bit `CRL Sign`.

Odvolané certifikáty jsou v CRL specifikovány podle svých sériových (tj. pořadových) čísel, takže pro běžného uživatele je CRL docela odtažitou záležitostí. CRL je určeno pro počítačové zpracování. V případě nepřímých CRL musí být odvolaný certifikát specifikován kromě sériovým číslem ještě názvem CA, tj. pomocí položky Vydavatel.

Tvar CRL specifikovala norma X.509. Na Internetu používáme CRL, které sice vycházejí ze specifikace CRL podle normy X.509 verze 2, ale opět je tato struktura pro Internet popsána v RFC-5280. Struktura CRL je obdobná struktuře certifikátu (Obrázek 5.3).

- První položkou CRL je položka Verze (*Version*), RFC-5280 předepisuje, že v CRL musí být použita a musí mít hodnotu 1 (tj. CRL dle X.509 verze 2).
- Položka Algoritmus podpisu (*Signature Algorithm*) obsahuje identifikátor algoritmů použitých pro elektronický podpis CRL (obdobně jako v certifikátu).
- Položka Vydavatel (*Issuer*) identifikuje, kdo tento CRL vydal (obdobně jako v certifikátu).

- Položka čas vydání CRL (*This Update*) specifikuje čas vydání tohoto CRL.



Obrázek 5.3 Struktura CRL

- Položka Předpokládaný čas vydání následujícího CRL (*Next Update*) určuje nejpozdější datum a čas vydání následujícího CRL. Následující CRL může být vydán dříve nebo v čas uvedený v této položce. Následující CRL nesmí být vydán později, než je datum a čas uvedené v této položce předchozího CRL.
- Položka Odvolané certifikáty (*Revoked Certificates)* obsahuje seznam odvolávaných certifikátů. Pro každý odvolávaný certifikát obsahuje sekvenci údajů:
  - Pořadové číslo odvolávaného certifikátu, které společně s položkou Vydavatel jednoznačně identifikuje odvolávaný certifikát (nejedná-li se o nepřímé CRL, pak se vydavatel bere z rozšíření položky odvolaných certifikátů).
  - Datum a čas, kdy držitel podal žádost o odvolání certifikátu, obsahuje položka Obdržení požadavku na odvolání certifikátu (*Revocation Date)*.
  - Volitelná položku rozšíření odvolávaného certifikátu obsahuje jednolivá rozšíření týkající se odvolání tohoto certifikátu.
- Nepovinná položka Rozšíření CRL (*CRL Extension)* obsahuje rozšíření CRL.

Všimněte si, že v CRL jsou dva typy rozšíření:

- Rozšíření položky CRL (*CRL Entry Extensions*), které obsahuje informace o odvolávání konkrétního certifikátu.
- Rozšíření CRL (*CRL Extensions*) – rozšíření CRL jako celku, které je obdobou rozšíření certifikátu.

Příklad výpisu CRL ve Windows je uveden v Příkladu 16.9.

5.2.1 Rozšíření CRL (celého)

Rozšíření CRL jsou obdobou rozšíření certifikátu. Tato rozšíření mohou být opět označena jako závažná; při jejich zpracování si zpracovávající software musí být vědom, co takové závažné rozšíření znamená.

Rozšíření CRL (celého)	**RFC-5280 předepisuje:**	**Význam**

Identifikátor klíče úřadu *(Authority Key Identifier)*	Musí být použito	Není závažné	Viz stejnojmenné rozšíření certifikátu
Alternativní jméno úřadu *(Issuer Alternative Name)*		Nemělo by být závažné	Viz stejnojmenné rozšíření certifikátu
Číslo seznamu CRL *(CRL Numer)*	Musí být použito	Není závažné	Certifikační autorita pomocí tohoto rozšíření čísluje vydávaná CRL. Číslování se provádí pomocí monotónně rostoucí sekvencí čísel. V případě, že zjistíme, že certifikát je na CRL, pak okamžitě dostaneme strach, jestli nebyl na předchozích CRL a my jsme jej přesto používali. Musíme tedy zjistit, které bylo předchozí CRL. To zjistíme právě podle rozšíření Číslo seznamu CRL. Předchozí CRL musí mít předchozí pořadí v sekvenci čísel přidělovaným CRL.
Indikátor rozdílového seznamu CRL *(Delta CRL Indicator)*	Musí být v rozdílovém CRL	**závažné**	Toto rozšíření indikuje, že se jedná o rozdílové CRL. Přírůstkové CRL neobsahuje úplné CRL, ale jen změny od předchozího úplného CRL. Přírůstkové CRL obsahuje také číslo úplného CRL, od kterého jsou přírůstky brány.
Vystavování distribučního místa *(Issuing Distribution Point)*		**závažné**	Na první pohled by se mohlo zdát, že toto rozšíření je identické s rozšířením certifikátu *CRL Distribution Points*. To je však omyl. Rozšíření Vystavování distribučního místa je závažným rozšířením a indikuje, že se nejedná o kompletní CRL vydavatele, ale o částečné CRL. Případně obsahuje odkazy na další dílčí částečná CRL.

5.2.2 Rozšíření položky CRL

Kromě samotného odvolání certifikátu mohou být zajímavé i další informace související s tímto odvoláním. Tyto informace mohou obsahovat rozšíření položky CRL. Tato rozšíření by neměla být označována jako kritická.

5.2.2.1 Důvod odvolání certifikátu (Reason Code)

Toto rozšíření identifikuje důvod odvolání certifikátu. CRL by měl obsahovat vždy konkrétní důvod odvolání certifikátu. Je však možné v odůvodněných případech toto rozšíření neuvést, což je stejné, jako by byl uveden kód pro „důvod nespecifikován“ (*unspecified reason*).

5.2.2.2 Instrukce pro případ pozastavení certifikátu (Hold Instruction Code)

Toto rozšíření indikuje instrukce pro případ pozastavení platnosti certifikátu.

5.2.2.3 Čas kompromitace soukromého klíče (Invalidity Date)

Toto rozšíření obsahuje datum a čas, kdy byla zjištěna ztráta či prozrazení soukromého klíče, jež je důvodem odvolání certifikátu.

5.2.2.4 Vydavatel certifikátu (Certificate Issuer)

Toto rozšíření položky nepřímého CRL obsahuje jedinečné jméno vydavatele odvolaného certifikátu. Pokud první položka CRL neobsahuje toto rozšíření, pak se jedná o přímé CRL. Pokud toto rozšíření neobsahuje další položka CRL, pak se za vydavatele bere týž vydavatel jako v předchozí položce.

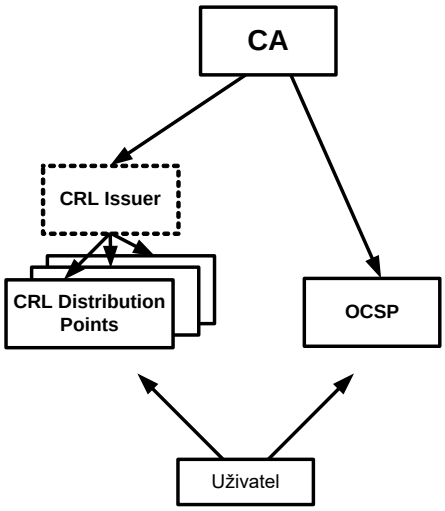
5.3 On Line zjišťování statusu certifikátu

Pokud uživatel používá certifikát pouze v jedné aplikaci (např. při styku s bankou), nejspíše okamžitě zkontaktuje provozovatele této aplikace a certifikát

si nechá v této aplikaci zablokovat. V tomto případě snad ani CRL nepotřebuje.

Jenže pokud uživatel používá certifikát k více účelům, pak často vyžaduje službu, která by o odvolání jeho certifikátu informovala ostatní uživatele co nejrychleji. Jako řešení se nabízí On Line služba organizovaná CA. Obecně nehovoříme jen o odvolání certifikátu ale o změně statusu certifikátu.

On Line služby jsou vesměs konstruovány na základě architektury klient/server. Uživatel, který chce certifikát použít, se On Line dotáže serveru poskytujícímu tyto služby na status certifikátu. Pokud server odpoví, že certifikát je platný, pak jej uživatel použije. Pokud server odpoví, že certifikát je odvolán, pak uživatel jeho použití zamítne. A pokud server odpoví, že status tohoto certifikátu je mu neznámý, pak je na uživateli (resp. jeho software) aby se rozhodl.



Obrázek 5.4 On Line služby jsou paralelními službami k CRL

Jiným aspektem je otázka jestli vůbec může CA tvrdit o nějakém certifikátu, že je odvolán, když není uveden na oficiálním CRL. Nebo On Line informace má snad jen informativní charakter? To si musí už ale vyřešit tvůrce certifikační politiky konkrétní CA. Na první pohled to vypadá jednoduše. Jenže stačí si uvědomit jak informace o tom, že certifikát byl odvolán, ale nebyl na CRL, bude vypadat po několika dnech odstupu a nebudeme mít On Line odpověď archivovánu. To pak budou dohady typu: A opravdu to tak bylo?

Protokolem určeným pro On Line zjišťování statusu certifikátu je např. protokol OCSP (*Online Certificate Status Protocol*). Serverem, který On Line status certifikátu poskytuje je OCSP server. CA deleguje pravomoc poskytovat On Line status svých certifikátů tím, že OCSP serveru vystaví speciální certifikát, který má v rozšíření Rozšířené použití klíče (*Extended Key Usage*) explicitně vyznačeno, že se jedná o certifikát OCSP serveru.

Pomocí protokolu OCSP lze řešit dva problémy:

- CRL vystavuje CA zpravidla s nějakou minimální periodou – tj. dávkově. V případě, že držitel certifikátu zjistí, že jeho soukromý klíč byl ztracen nebo odcizen, pak mu mechanismus CRL nebude příliš vyhovovat, protože do vydání dalšího CRL může být jeho soukromý klíč zneužit. Protokolem OCSP alespoň teoreticky může certifikační autorita poskytovat informace o odvolání certifikátu rychleji než pomocí CRL.
- CRL může být značně rozsáhlá datová struktura, jejíž stažení a zpracování může značně zdržovat. OCSP dotaz a následná odpověď tak leckdy mohou být rychlejší a méně zdržovat uživatele. OCSP pak nenahrazuje CRL, ale zrychluje zpracování certifikátu.

5.4 Platnost certifikátu k uvedenému datu

Pokud máme např. digitálně podepsaný dokument a chceme si ověřit platnost jeho elektronických podpisů, pak nás bude zajímat platnost příslušných certifikátů nikoliv teď, ale v okamžiku podepsání dokumentu. Potřebujeme tedy On Line dotaz na status certifikátu doplnit o datum, ke kterému chceme zjistit platnost certifikátu.

Protokol OCSP verze 1 však není určen pro řešení problému platnosti certifikátu ke konkrétnímu datu v minulosti. V připravované verzi 2 se o této eventualitě uvažuje.

5.5 Vzdálené ověřování platnosti certifikátu

CRL mohou být značně rozsáhlá a jejich stahování a následné zpracování může značně zaměstnat náš počítač. Navíc pokud chceme ověřit platnost certifikátu, pak musíme ověřit platnost všech certifikátů v řetězci certifikátů až k důvěryhodné kotvě. Tj. musíme nejprve získat všechny certifikáty až k důvěryhodné kotvě a pak zjistit status všech certifikátů.

Tak jako v úřadech už každý uživatel nemívá vlastní tiskárnu, ale pro skupinu uživatelů je jeden printserver, který je rychlejší a efektivnější, co kdybychom měli také server, který by za nás verifikoval certifikáty. Ocenili by to jistě tvůrci miniaturních počítačů,

bylo by to výhodné i v rámci intranetu – poskytli bychom jeden server, který by příslušné informace zjišťoval a poskytovat centrálně ostatním PC.

RFC-3379 zavádí dvě strategie (dva protokoly) jak takovou situaci řešit:

- DPV (*Delegated Path Validation*) protokol umožní plně delegovat verifikaci certifikátu na DVP server. Server provádí verifikaci sám a klientovi vrací výsledek verifikace.
- DPD (*Delegated Path Discovery*) protokol. DPD server slouží k tomu, aby poskytoval veškeré informace nutné pro ověření certifikátu. Tj. klient požádá DPD server o informace, DPD server je ve své odpovědi poskytne a klient si sám lokálně provede verifikaci certifikátu.

Toto RFC specifikuje sadu požadavků, které by nový protokol(y) měl naplňovat. V rámci pracovní skupiny IETF PKIX probíhají práce na specifikace tohoto protokolu s názvem Standard Certificate Validation Protocol (SCVP).

6 Certifikační cesta a důvěryhodné kotvy

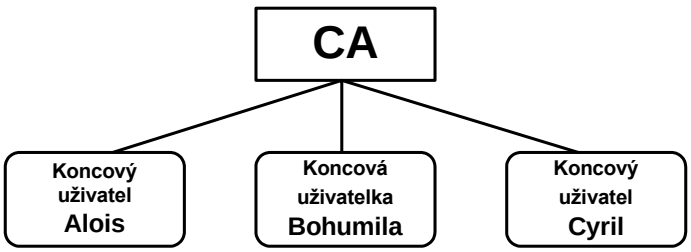
Dosud jsme předpokládali, že máme samostatnou (*stand alone*) certifikační autoritu, která vystavuje certifikáty svým koncovým uživatelům (Obrázek 6.1). Tato představa předpokládá, že tato certifikační autorita má kořenový (*self signed*) certifikát. Kořenový certifikát certifikační autority má několik zvláštností:

Je podepsán soukromým klíčem příslušejícím k veřejnému klíči uvedeném v témže certifikátu CA (Obrázek 6.2).

- Pokud by se ověřoval elektronický podpis kořenového certifikátu, pak by se využil veřejný klíč z téhož certifikátu.
- Ověření elektronickýho podpisu kořenového certifikátu tedy nepřinese nic než kontrolu integrity datové struktury certifikátu. Elektronický podpis kořenového certifikátu má tedy z pohledu jeho ověření nanejvýš hodnotu jako otisk.
- Podvrhnout kořenový certifikát je tudíž velice snadné. Útočníkovi stačí vygenerovat libovolnou dvojici veřejný/soukromý klíč a k takto vygenerovanému veřejnému klíči vytvoří kořenový certifikát s libovolným obsahem.
- Kořenové certifikáty musí být proto distribuovány důvěryhodnou cestou. Takou cestou je např. osobní předání kořenového certifikátu s vydaným certifikátem koncového uživatele na registrační autoritě (RA) či distribuce kořenového certifikátu v rámci distribuce software (např. distribuce operačního systému).
- Kořenovým certifikátům tedy důvěřujeme, protože jsme je získali z důvěryhodného zdroje. Pokud obdržíme kořenový certifikát jinou cestou, pak mu explicitně důvěřovat nemůžeme a musíme si ověřit jeho důvěryhodnost. Nejběžnějším postupem ověření důvěryhodnosti takového certifikátu je, že se jinou cestou (např. telefonicky) spojíme s důvěryhodnou osobou (např. operátorem CA) a dotážeme se jej na důvěryhodnost certifikátu. Jak to uděláme? Požádáme jej, aby nám recitoval otisk z tohoto certifikátu, a recitovaný tisk pak porovnáme s námi spočteným otiskem ověřovaného certifikátu. Nutno dodat, že je třeba, aby obě strany použily stejný algoritmus pro výpočet otisku. A také je třeba dodat, že mnohdy není třeba vyžadovat celý otisk, ale stačí jen některé cifry z otisku (fragment otisku).
- Certifikátům CA (nebo veřejným klíčům) jejichž platnost ověřujeme jinou cestou a tudíž jim explicitně důvěřujeme, říkáme **důvěryhodné kotvy**. Kořenový certifikát, který jsme ověřili jinou cestou, je příkladem důvěryhodné kotvy.
- Při verifikaci certifikátu důvěryhodným kotvám věříme, a tudíž je neověřujeme – pouze využijeme údaje z nich (zejména informace týkající se veřejného klíče). Důvěryhodná kotva tak není součástí tzv. certifikační cesty (viz kap. 8).
- Jelikož důvěryhodná kotva není součástí certifikační cesty, tak v případě, že má tvar certifikátu (nejenom kořenového) se z tohoto certifikátu berou v úvahu jen některé údaje. Např. většina rozšíření certifikátu se v případě důvěryhodné kotvy ignoruje!
- Jiným způsobem distribuce důvěryhodné kotvy je její distribuce jako důsledek, že naše je PC součástí podnikové sítě např. na bázi ActiveDirecotory, pak certifikáty důvěryhodných CA mohou být distribuovány skrze ActiveDirectory. Např. skrze skupinovou politiku (*Group Policy*) či NTAuthSore.
- Součástí novějších operačních systému Microsoft je komponenta „*Update Root certificates*“. Pokud si ji nainstalujete, pak vám Microsoft pravidelně přes Internet aktualizuje seznam důvěryhodných kotev (viz též paragraf 7.4.1). Každá mince má však dvě strany. Na intranetu bude taková funkčnost patrně nevítaná.

6.1.1 Podvržení kořenového certifikátu

Pokud Bohumila pošle Aloisovi elektronickým podpisem stvrzený rozchod (k podpisu přidá i certifikát kořenové CA jež jej vydala), pak Alois nesmí mechanicky verifikovat certifikát Bohumily včetně kořenového certifikátu CA zaslaného Bohumilou, ale musí zjistit, zdali se Bohumilin zaslaný kořenový certifikát



Obrázek 6.1 Samostatná CA vydávající certifikáty koncovým uživatelům

shoduje s kořenovým certifikátem, který má uložen mezi svými důvěryhodnými kotvami!

Pokud by totiž takovou verifikaci neprovedl, pak by také bylo možné, že zprávu vytvořil žárlivý Cyril. Jak by to udělal? Jednoduše:

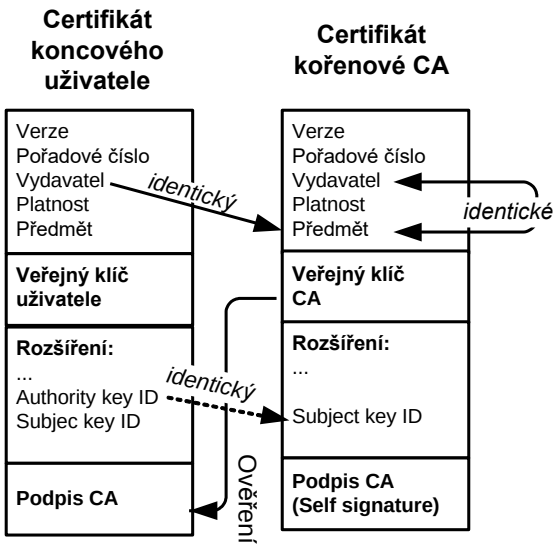
12. Vygeneroval by dvojici veřejný/soukromý klíč.
13. Tuto dvojici by Cyril podvrhl za veřejný/soukromý klíč CA (vytvořil by podvržený certifikát kořenové CA):
    - Do podvrženého certifikátu by z certifikátu Bohumiliny oblíbené CA zkopíroval veškeré údaje CA kromě veřejného klíče.
    - Jako veřejný klíč by Cyril použil právě vygenerovaný veřejný klíč.
    - Tvorbu podvrženého certifikátu CA by završil elektronickým podpisem certifikátu podvržené CA, k němuž by Cyril použil právě vygenerovaný soukromý klíč.
14. Nyní by Cyrilovi již nic nebránilo podvrhnout i certifikát Bohumily.
    - Vygeneroval by další dvojici veřejný/soukromý klíč, která mu umožní vytvořit podvržený certifikát Bohumily.
    - Do podvrženého certifikátu Bohumily by zkopíroval veškeré identifikační údaje z pravého certifikátu Bohumily kromě veřejného klíče.
    - Do certifikátu by vložil vygenerovaný (podvržený) veřejný klíč.
    - Certifikát Bohumily podepíše podvrženou CA.
    - Nyní již může vytvořit podvržený dokument, který bude verifikovatelný podvrženým certifikátem Bohumily.

6.1.1.1 *Ověření certifikátu Bohumily*

Jak jsme se již zmínili, tak kořenový certifikát CA se verifikuje jinou cestou – je totiž důvěryhodnou kotvou. Nadále tedy budeme předpokládat, že kořenový certifikát CA je důvěryhodnou kotvou, o které nepochybujeme.

Verifikace certifikátu Bohumily má tři polohy:

- Samotná verifikace certifikátů vůči důvěryhodné kotvě. Je třeba mít na paměti, že naše CA může mít více certifikátů. Tj. musí být shoda nejenom v tom, že položka Vydavatel (*Issuer*) z certifikátu Bohumily je shodná s položkou



Obrázek 6.2 Verifiace certifikátu koncového uživatele vydaného CA mající kořenový certifikát

  Předmět (*Subject*) v certifikátu CA, ale zejména hodnota rozšíření Identifikátor klíče úřadu (*Autority Key Identifier*) v certifikátu Bohumily musí být shodná s rozšířením Identifikátor klíče předmětu (*Subject Key Identifier*) certifikátu CA[*)]. Tato verifikace je podmnožinou verifikace certifikátu v řetězci certifikátů, které je věnována následující kapitola, proto se bližšími detaily zde již zabývat nebudeme.
- Ověření zdali certifikát nebyl odvolán. To je možné pomocí CRL nebo pomocí On Line metod.
- Je možné certifikát využít v konkrétní aplikaci:
  - Je pro konkrétní aplikaci použitelný příslušný asymetrický algoritmus?
  - Umožňují využití certifikátu jeho rozšíření: Použití klíče (Key Usage), Rozšířené použití klíče (Extended Key Usage) atd.
  - Umožňuje to certifikační politika vyznačená v certifikátu?

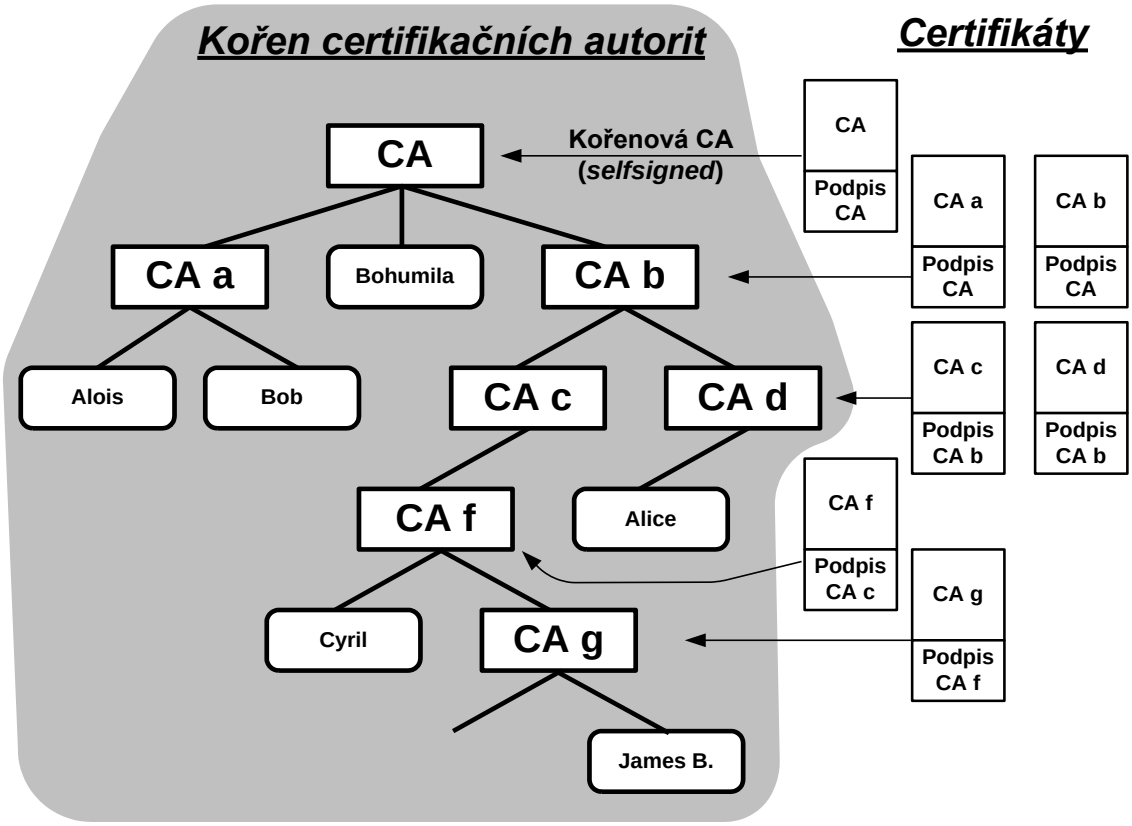
6.1.2 Strom certifikačních autorit

Realita však může být složitější. Naše certifikační autorita CA může být jen kořenem ve stromu certifikačních autorit.

Certifikační autorita CA může vydat kromě certifikátů koncových uživatelů i certifikáty podřízeným certifikačním autoritám CAa a CAb. A obdobně např. certifikační autorita CAb může vytvořit své podřízené certifikační autority CAc a CAd tak, že jim vydá certifikáty. Výsledkem je stromová struktura certifikačních autorit (Obrázek 6.3).

[*)] Jinou variantou je do rozšíření Identifikátor klíče úřadu uložit jedinečné jméno CA a sériové číslo certifikátu CA, kterým se ověřovaný certifikát ověřuje. V takovém případě certifikát CA nemusí ani mít rozšíření Alternativní jméno předmětu.

Jak vypadá certifikát podřízené certifikační autority? Nejdůležitější změnou oproti certifikátu kořenové CA je v tom, že tento certifikát má různé hodnoty v položkách Vydavatel (*Issuer*) a Předmět (*Subjekt*). X.509 označuje takové certifikáty jako křížové certifikáty (*cross-certificate*). Nesmí se však termín křížový certifikát zaměňovat s termínem křížová certifikace certifikačních autorit, o kterém bude zmínka dále (křížová certifikace je oboustranná, tj. vyžaduje dva křížové certifikáty – každý v jednom směru).



Obrázek 6.3 Strom certifikačních autorit

Na první pohled je tedy křížový certifikát obdobný certifikátu koncového uživatele. Jenže při podrobnějším zkoumání zjistíme, že křížový certifikát se značně liší zejména v rozšířeních certifikátu. Proč? Důvod je prostý. Tím, že někomu vydáme křížový certifikát, tak mu dáváme do ruky moc vydávat další certifikáty. A pokud nějakým způsobe ručíme za vydávané certifikáty, tak přeneseně budeme ručit i za certifikáty vydané osobou, které jsme vydali křížový certifikát. Jelikož se trochu bojíme, aby nezvlčil a vydával certifikáty zodpovědně, tak se jej budeme snažit omezit v neuváženém vydávání certifikátů.

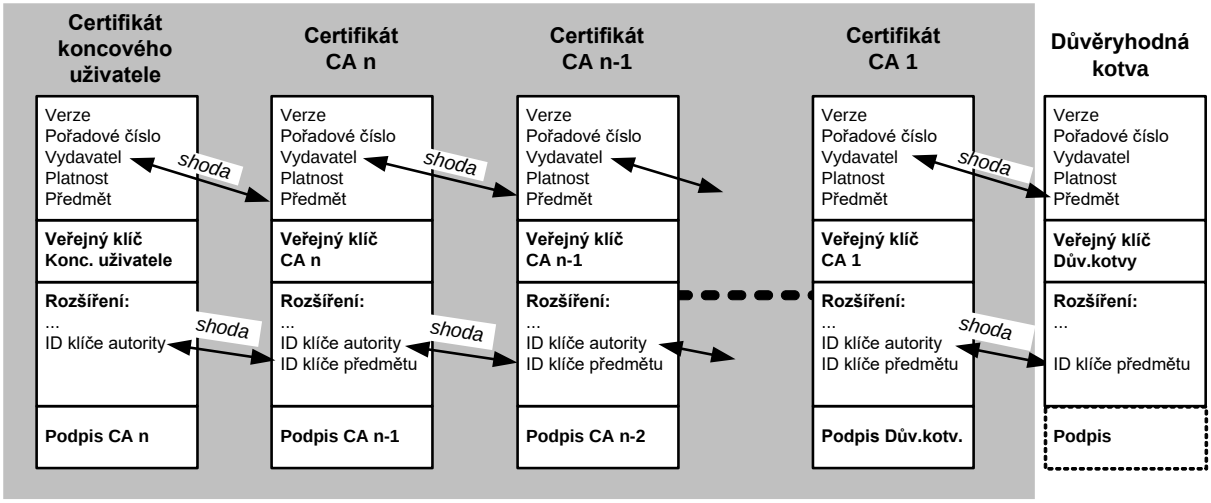
Pomineme-li rozšíření certifikátu Použití klíče (*Key Usage*), tak držitele křížového certifikátu můžeme omezovat ve vydávání certifikátů zejména promyšleným využitím rozšíření certifiktu, která mají v názvu slovo „omezení“ (*constrain*). Pomocí rozšíření Základní omezení (*Basic Constrains*) jej můžeme omezovat ve vydávání dalších křížových certifikátů, pomocí rozšíření Omezení jmen (*Name Constranins*) jej můžeme omezit na vydávání certifikátů subjektům mající např. jména pouze z konkrétních DNS domén atd.

6.1.2.1 Řetězec certifikátů

V případě, že máme strom certifikačních autorit, pak ověření certifikátu koncového uživatele je složitější než v případě kořenové certifikační autority. Je třeba provést nejen ověření certifikátu koncového uživatele, ale i ověření certifikátu certifikační autority, která tento certifikát vydala. Pakliže i toto certifikační autorita má křížový certifikát, pak musíme verifikovat i ten. Neverifikuje pak jeden certifikát, ale řetězec certifikátů (Obrázek 6.4).

Řetězec certifikátů je zakončen důvěryhodnou kotvou. Důvěryhodná kotva bývá zpravidla ve tvaru kořenového certifikátu. Teoreticky můžeme i jako důvěryhodnou kotvu využít i samotný veřejný klíč (včetně jeho případných parametrů a jména jeho vydavatele).

Máme-li k dispozici množinu certifikátů, pak prvním problémem je vytvořit řetězec certifikátů až k důvěryhodné kotvě. Tvorbu řetězce certifikátů zpravidla začínáme od certifikátu koncového uživatele. K němu hledáme certifikát certifikační autority, která jej vydala tak, že najdeme všechny certifikáty,



Obrázek 6.4 Certifikační cesta. Při jejím sestavování postupujeme „z leva do zprava“, ale při jejím ověřování postupujeme „zprava do leva“!

jejichž položka Předmět (*Subjekt*) je identická s položkou Vydavatel (*Issuer*). Takových certifikátů může být i více. Mezi nimi vybereme ten, jehož hodnota rozšíření Identifikátor klíče předmětu (*Subject Key Identifier*) je shodná s hodnotou rozšíření Identifikátor klíče úřadu (*Autority Key Identifier*) ověřovaného certifikátu. Pak zjistíme, jestli certifikát této certifikační autority náhodou nemáme mezi důvěryhodnými kotvami. Pokud ano, pak již nehledáme další prvek řetězce certifikátů, pokud nikoliv, pak postupujeme obdobně dále, až dosáhneme důvěryhodné kotvy.

Pokud jsme si postavili řetězec certifikátů, pak přikročíme k ověřování (verifikaci) certifikátu. Při ověřování certifikátu postupujeme zpět od důvěryhodné kotvy přes všechny křížové certifikáty až k ověřovanému certifikátu. Pro každý certifikát v řetězci musíme ověřit, jestli předchozí (nadřízená) certifikační autorita byla oprávněna vydat v řetězci uvedený certifikát. Jestli některý certifikát z řetězce certifikátů není uveden na CRL atd. Pakliže selže ověření jednoho certifikátu v řetězci certifikátů, pak selže ověřování ověřovaného certifikátu a ověřovaný certifikát musí být odmítnut. Verifikaci certifikátu v řetězci certifikátů se budeme podrobně věnovat v následující kapitole.

Ověříme-li platnost křížových (mezilehlých) certifikátů v certifikační cestě, pak informaci o jejich ověření si můžeme uložit a po jistou dobu těmto certifikátům důvěřovat obdobně jako důvěryhodné kotvě. Nebo naopak je můžeme považovat za odvolané. Tento mechanismus může pak značně zrychlit ověřování certifikátů. Je však rozdíl mezi důvěryhodnou kotvou tvořenou kořenovým certifikátem a těmito důvěryhodnými mezilehlými, křížovými certifikáty. Rozdíl spočívá v množství rozšíření certifikátu, které mají. V případě důvěryhodné kotvy se totiž jiná rozšíření než Alternativní jméno předmětu nevyužijí, kdežto rozšíření z mezilehlých certifikátů se zpravidla ověřují.

6.1.3 Vzájemná důvěra mezi certifikačními autoritami

Může se však stát, že dva uživatelé mají certifikáty vydány od různých certifikačních autorit, které navíc nejsou součástí téhož stromu certifikačních autorit (Obrázek 6.5).

Jak pak může Alois důvěřovat např. elektronicky podepsaným zprávám od Bohumily, když sám důvěřuje jen certifikační autoritě CA1 a Bohumila důvěřuje jen certifikační autoritě CA2?

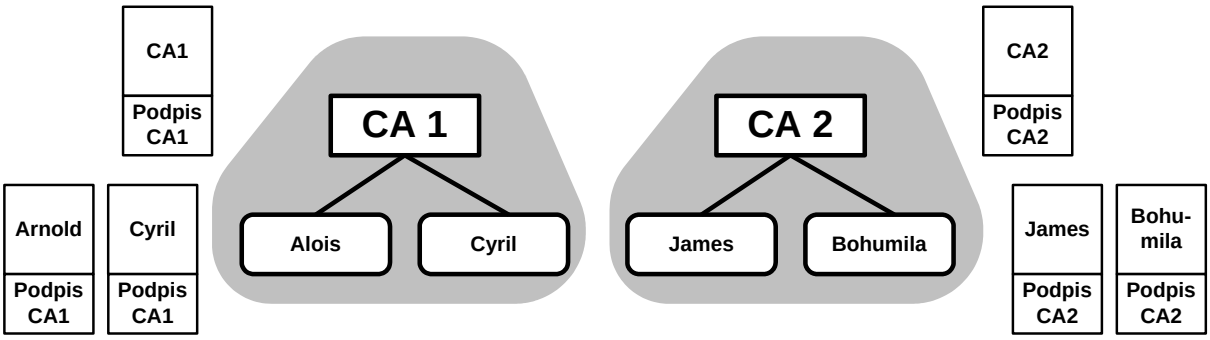
Řešení tohoto problému máme několik:

- Alois si řekne: když Bohumila důvěřuje CA2 , tak já ji také budu důvěřovat a doplní si certifikát CA2 mezi své důvěryhodné kotvy.
- Vzájemní křížová certifikace CA1 a CA2
- Kořenová certifikační autorita tvořící most (*bridge*) mezi stromy.
- CTL (*Certificate Trusted List*), tzv. „Seznam důvěryhodných certifikátů“.

6.1.3.1 *Křížová certifikace*

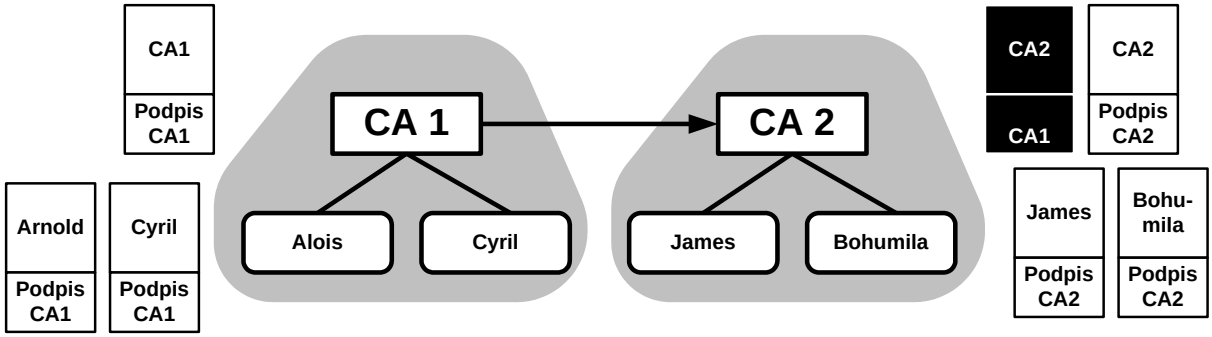
Doposud jsme předpokládali, že certifikační autority tvoří stromovou strukturu. Avšak vše může být ještě komplikovanější. Dvě certifikační autority, které nejsou zařazeny do téže stromové struktury, si mohou vzájemně podepsat své certifikáty – provést křížovou certifikaci. (Pro úplnost je třeba dodat, že křížově se teoreticky mohou certifikovat i CA patřící do společného stromu – to má však význam, když je strom příliš košatý a CA leží na velmi vzdálených větvích stromu; pak lze křížovou certifikací jejich vzdálenost zmenšit. Je to ale spíše jen teoretická úvaha, kterou můžeme ještě doplnit o skutečnost, že v takovém případě by v certifikační cestě mohly vznikat dokonce smyčky … .)

Na obrázku Obrázek 6.8 je vidět jak se nově vydaný certifikát CA2 prakticky uplatní. Aby James, důvěřující podobně jako Bohumila certifikační autoritě CA2, verifikoval certifikát Bohumily, pak vytvoří řetězec certifikátů bez nově CA1 vydaného certifikátu CA2. Jamesovi takový řetězec certifikátů stačí, protože pro něj je kořenový certifikát CA2 důvěryhodnou kotvou.



Obrázek 6.5 Dva stromy certifikačních autorit

Jenže pro Aloise není kořenový certifikát CA2 důvěryhodnou kotvou, pro něj je důvěryhodnou kotvou pouze kořenový certifikát CA1. Ale pomocí certifikátu CA2 vydaného CA1 je schopen vytvořit řetězec certifikátu až k jeho důvěryhodné kotvě. Takže i Alois nyní může důvěřovat i certifikátu Bohumily.



Obrázek 6.6 CA1 vydala CA2 další certifikát certifikační autority

Avšak vraťme se na počátek k vysvětlení podstaty křížové certifikace (Obrázek 6.6). Aby Alois mohl důvěřovat zprávám podepsaným Bohumilou a nemusel si certifikát CA2 uložit mezi své důvěryhodné kotvy, tak si CA2 kromě svého kořenový certifikátu nechala vydat certifikát od CA1.

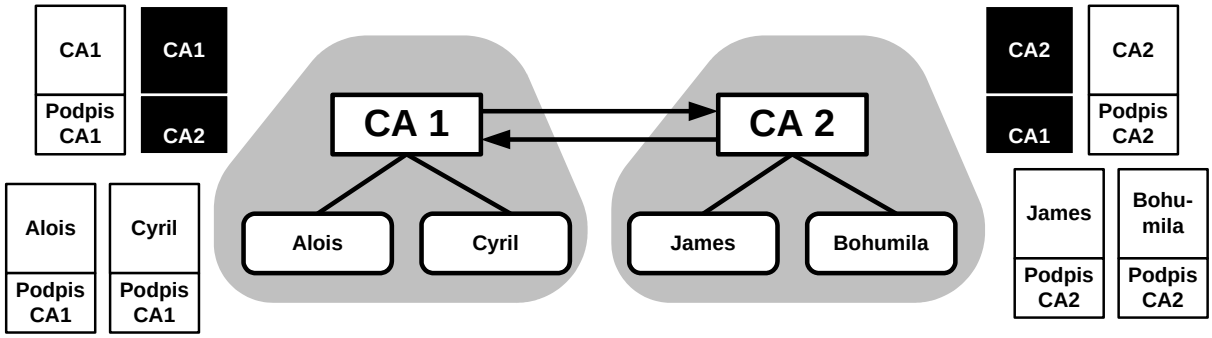
Takže CA2 nyní má dva certifikáty certifikační autority: (1) kořenový, (2) vydaný CA1. Technicky to CA2 provedla jednoduše. Když si vystavovala svůj kořenový certifikát, tak k tomu vytvořila příslušnou žádost o certifikát. Nyní tutéž starou žádost uplatní u CA1.

Jenže v případě, že Alois pošle Bohumile podepsanou odpověď, tak Bohumila není schopna nalézt řetězec, důvěryhodné kotvě. Je tedy nutné, aby nejenom CA2 si nechala vydat certifikát od CA1 ale, ale naopak, aby i CA1 si nechala vydat certifikát od CA2 (Obrázek 6.7). Říkáme, že CA1 se s CA2 vzájemně křížově certifikovaly. Výsledkem je, že máme čtyři certifikáty:

- CA1 podepsaný CA1, tj. kořenový certifikát CA1.
- CA2 podepsaný CA2, tj. kořenový certifikát CA2..
- CA1 podepsaný CA2, tj. křížový certifikát.
- CA2 podepsaný CA1, tj. křížový certifikát.

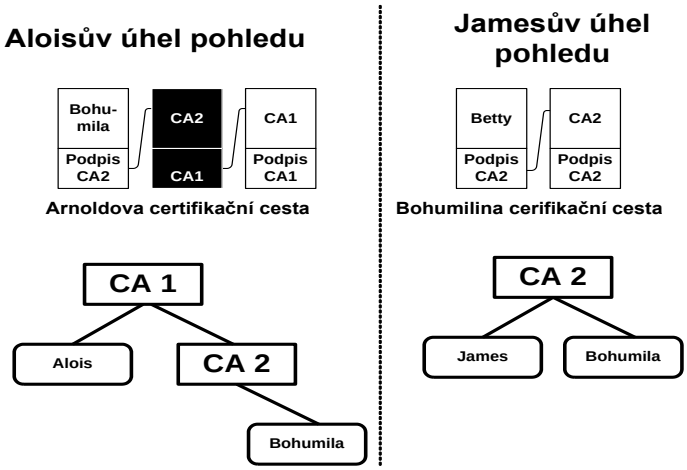
Nyní si mohou Alois s Bohumilou oboustranně důvěryhodně komunikovat. Pro ověření certifikátů bude každý z nich vytvářet jiné řetězce certifikátů (každý ke „své“ důvěryhodné kotvě), ale komunikace bude možná.

V případě digitálně podepsaných zpráv odesilatelé zpravidla neposílají samotnou digitálně podepsanou zprávu, ale též certifikáty, o nichž se domnívají, že příjemce je bude potřebovat pro verifikaci elektronického podpisu (tj. bude je potřebovat k sestavení certifikační cesty). Odesilatel ale neví, jaké má adresát důvěryhodné kotvy, tak mu ke zprávě zabalí množinu certifikátů. Je pak na adresátovi aby byl schopen z prvků této množiny vytvořit řetězec certifikátů až k některé z jeho důvěryhodných kotev. Adresát si však může potřebné křížové certifikáty obstarat i jinou cestou (např. využitím rozšíření Přístup k informacím úřadu (*Authority Information Access*) či obecně na LDAP serverech certifikačních autorit).



Obrázek 6.7 Vzájemná křížová certifikace certifikačních autorit CA1 a CA2

V případě, že se křížově certifikují dvě certifikační autority, tak křížová certifikace na obrázku vypadá jednoduše. Jenže pokud chcete vzájemně křížově certifikovat 10 certifikačních autorit, pak množství křížových certifikátů je značné a problém je netriviální. A pak se jednoho dne jako na potvoru stane, že potřebujete obnovit certifikát některé z těchto certifikačních autorit a máte o zábavu postaráno.

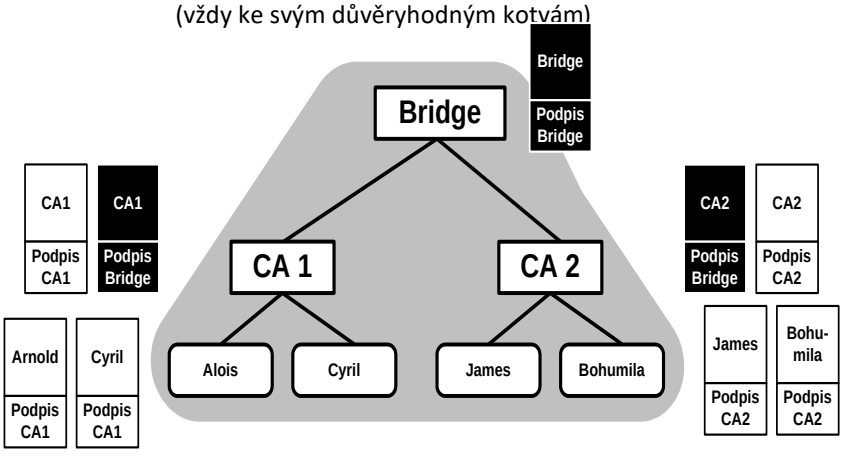


Obrázek 6.8 Alois a Bohumila vytvoří různé řetězce certifikátů (vždy ke svým důvěryhodným kotvám)

6.1.3.2 Most certifikačních autorit (Bridge)

Vzájemná křížová certifikace může proběhnout po vzájemné dohodě dvou certifikačních autorit. Problém je ale v případě, když by se mělo vzájemně křížově certifikovat větší množství certifikačních autorit. Pokud chceme propojit více certifikačních autorit, pak křížová certifikace není tím pravým ořechovým. Možným řešením je most (*bridge*).

Certifikační autority, které si chtějí vzájemně důvěřovat, společně vybudují kořenovou certifikační autoritu – mostní kořenovou certifikační autoritu - *bridge* (Obrázek 6.9).



Obrázek 6.9 Most (bridge)

7 Ověřování platnosti certifikátu

V předchozí kapitole jsme sestavovali certifikační cesty od certifikátu, jež chceme ověřovat až k důvěryhodné kotvě. Všimli jsme si, že pro konkrétní certifikát můžeme teoreticky sestavit i více certifikačních cest. Dokonce v certifikačních cestách se mohou vyskytovat i cykly[*)]. Je pak na konkrétní implementaci jak se k této problematice postaví.

Nyní je našim cílem již sestavenou certifikační cestu ověřit. Tj. budeme se zabývat ověřováním (verifikací) konkrétní sestavené certifikační cesty.

Mlčky předpokládáme, že chceme ověřit, zdali je certifikát platný právě teď – v tomto okamžiku. Jinou úlohou je potřebujeme-li ověřit, že certifikát byl platný k nějakému okamžiku v minulosti. To je podstatně složitější problém, který vyvstane v případě ověřování pravosti archivovaných elektronicky podepsaných dokumentů.

7.1.1 Ověřování cesty začíná od důvěryhodné kotvy!

Při sestavování certifikační cesty jsme postupovali od ověřovaného certifikátu. V případě ověřování pak postupujeme v opačném směru - od důvěryhodné kotvy až k ověřovanému certifikátu. Jelikož důvěryhodné kotvě věříme a tudíž jí nemusíme ověřovat, tak postupně ověřujeme platnost prvního certifikátu pod důvěryhodnou kotvou, pak druhého, až dojdeme k certifikátu, jež platnost máme ověřit. Šťouralové by tedy podotkli, že nadpis tohoto odstavce (Ověřování cesty začíná od důvěryhodné kotvy) není zcela pravdivý, protože přece důvěryhodná kotva není součástí certifikační cesty!

Představa, že pro ověření certifikátu nejprve sestavíme certifikační cestu a teprve až ji máme sestavenu, tak začneme s jejím ověřováním je představa školská, se kterou se též setkáváme ve standardech. Jelikož sestavení a ověření certifikační cesty často trvá vteřiny nebo i desítky vteřin což obtěžuje uživatele, tak implementátoři se snaží tento čas urychlit. Implementátoři již při sestavování certifikační cesty startují ověřování všeho, co je možné ověřit již při sestavování cesty (např. je certifikát publikován na CRL?). Standard pak výslovně říká, že implementován může být i jiný algoritmus než že se nejprve sestaví certifikační cesta, která se až následně ověřuje, ale takový jiný algoritmus musí vést k témuž výsledku.

Zde je pak možné nalézt i odpověď na otázku: jak je možné, že ověřování certifikátů je ve Windows 2003 rychlejší než ve Windows 2000 – Microsoft vytvořil efektivnější ověřovací algoritmus (v jeho terminologii: rychlejší ověřovací stroj).

7.1.2 Ověřujeme certifikační cestu

Certifikát ověřujeme proto, abychom ověřili, že veřejný klíč uvedený v certifikátu je platný a opravdu náleží držiteli uvedeném v předmětu certifikátu. A dále, že tento klíč může být využit k danému účelu. Proto jsme vytvořili certifikační cestu až důvěryhodné kotvě a nyní ji budeme ověřovat. Certifikát vydaný důvěryhodnou kotvou označujeme jako první certifikát v certifikační cestě. Certifikát vydaný prvním certifikátem označujeme jako druhý certifikát v certifikační cestě atd. Důvěryhodná kotva se často označuje jako certifikát číslo 0.

Důvěryhodná kotva pravděpodobně bude ve tvaru kořenového certifikátu. Z tohoto certifikátu se pro ověřování vezme:

- Jméno vydavatele
- Veřejný klíč včetně algoritmu veřejného klíče a případných parametrů veřejného klíče.

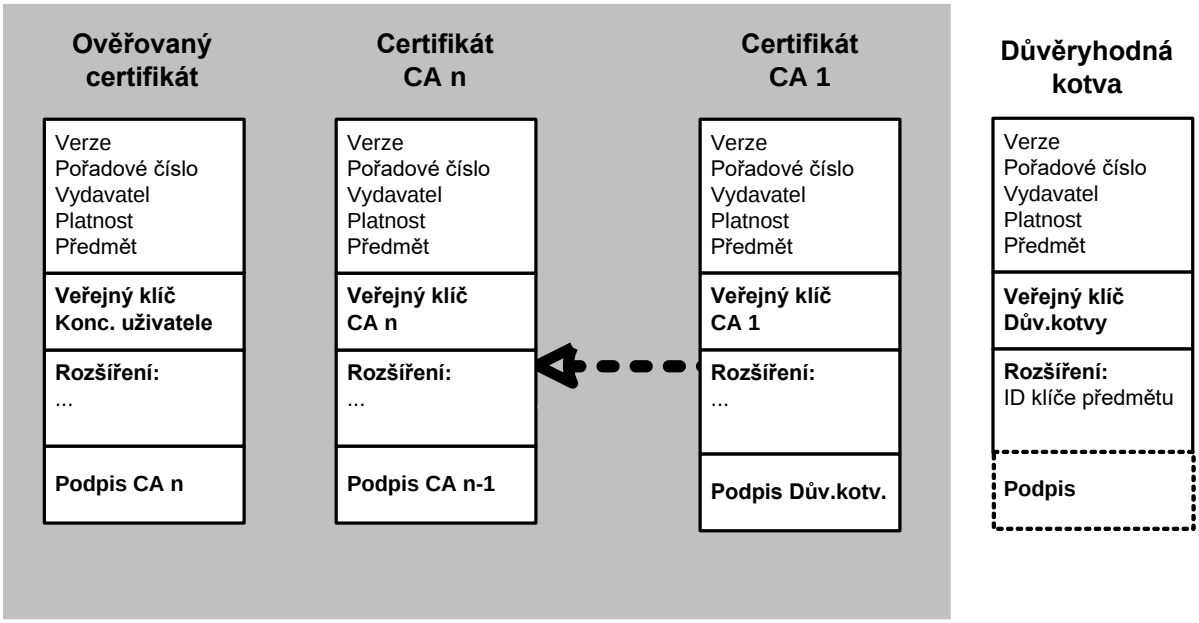
Jelikož ostatní informace se v důvěryhodné kotvě při ověřování certifikátu ignorují, tak důvěryhodné kotvy, pokud vůbec jsou ve tvaru certifikátu, pak zpravidla obsahují minimum rozšíření certifikátu. Pokud vydavatel důvěryhodné kotvy vydal více certifikátů se stejným předmětem, pak jediným rozšířením majícím reálným efekt je rozšíření Identifikátor klíče předmětu, které má efekt pro sestavení certifikační cesty (nikoliv pro ověření certifikátu).

Do procesu ověřování tak vstupuje:

- Sestavená certifikační cesta.
- Důvěryhodná kotva.
- Aktuální čas.
- Inicializační informace týkající se cerifikačních politik v případě, že si při ověřování certifikátu hodláme hrát na certifikační politiky.

Využití mnohých rozšíření certifikátu je volitelné. Takže pokud aplikace tato rozšíření nepodporuje, pak je nemusí využívat a jejich verifikaci může přeskočit. Nicméně pokud je nepodporované rozšíření označeno jako závažné, pak certifikát musí být odmítnut.

[*)] Pokud při sestavování certifikační cesty je vyžadován certifikát, který v cestě již je (tj. jedná-li se o náznak cyklu), pak zpravidla se v sestavování takové cesty již nepokračuje a hledá se jiná certifikační cesta.



Obrázek 7.1 Ověřování certifikační cesty probíhá od důvěryhodné kotvy k ověřovanému certifikátu

Při ověřování certifikátu půjdeme krok po kroku po certifikační cestě od certifikátu číslo 1 až k certifikátu, jež chceme ověřit. Představme si, že ověřujeme certifikát číslo ***x*** v certifikační cestě, pak (blíže viz RFC-5280):

- Ověříme, že vydavatel certifikátu je předmětem certifikátu ***x-1***.
- Ověříme elektronický podpis certifikátu ***x*** pomocí certifikátu ***x-1***.
- Ověříme, že aktuální čas padne do období platnosti certifikátu.
- Ověříme, že certifikát nebyl odvolán (viz následující odstavec).
- Ověříme, že jména uvedená v předmětu certifikátu a v rozšíření Alternativní jména předmětu odpovídají omezením vypívajícím z rozšíření Omezení jmen v předchozích certifikátech certifikační cesty.
- Verifikují se rozšíření spojená s certifikačními politikami (viz popis jednotlivých rozšíření certifikátu v kap. 15).

Pokud neselhalo ověřování žádného certifikátu certifikační cesty až k ověřovanému certifikátu, tak certifikát je platný.

7.1.3 Byl certifikát odvolán?

Informace o tom byl-li certifikát odvolán, můžeme získat z CRL nebo jinou cestou (např. pomocí OCSP protokolu). Kde (na jakém URI) tuto informaci získat je uvedeno v následujících rozšířeních ověřovaného certifikátu :

- V rozšíření Distribuční místa seznamu odvolaných certifikátů (*CRL Distribution Points*) jsou uvedeny URI na kterých je publikováno CRL.
- V rozšíření Nejčerstvější CRL (*Freshest CRL*) jsou uvedeny URI na kterých je publikováno přírůstkové CRL (*Delta CRL*).
- V rozšíření Přístup k informacím úřadu (*Auhtority Info Access*) může být publikováno i URI na kterém naslouchá OCSP server.

V případě ověřování, není-li certifikát publikován na CRL musíme mít k dispozici:

- Aktuální CRL.
- Ověřovaný certifikát, ze kterého vezmeme: jméno vydavatele (*issuer*) a číslo certifikátu jako identifikace certifikátu pro jeho vyhledávání v CRL.
- Informaci o tom budeme-li využívat přírůstková CRL (*Delta CRL*). Tato informace je signalizována pomocí rozšíření CRL, které se jmenuje Indikátor rozdílového seznamu CRL.

Nyní můžeme začít s ověřováním, nebyl-li certifikát uveden na CRL:

15. Získáme neprošlé CRL (i přírůstkové CRL). Proto postupně procházíme lokální vyrovnávací paměť (*cache*), kontaktujeme všechna URL uvedená v rozšíření Distribuční místa seznamu odvolaných certifikátů v certifikátu, až získáme CRL jehož položka Příští aktualizace (*Next Update*) obsahuje pozdější čas než je aktuální čas. Pozor (!): výsledkem je, že nemusíme získat aktuální CRL, ale neprošlé CRL!

16. Verifikuje, zdali CRL bylo vydáno vydavatelem odpovědným za vydávání CRL. Tj. v případě přímého CRL musí být vydavatel ověřovaného certifikátu a vydavatel CRL týž.
17. V případě přírůstkového CRL:
    - Opět verifikujeme, přísluší-li přírůstkové CRL k úplnému CRL, které máme k dispozici a jež využijeme ke zpracování.
    - Verifikuje, zdali obsah rozšíření Identifikátor klíče úřadu se shoduje v přírůstkovém CRL se shoduje s obsahem téhož rozšíření v úplném CRL.
18. V případě omezeného CRL pak ověříme, jsou-li důvody pro které je toto omezené CRL vydáno vyhovující důvodům ověřovaného certifikátu. Např. pokud ověřujeme certifikát certifikační autority, pak omezené CRL musí obsahovat odvolané certifikáty všech CA.
19. Vytvoříme certifikační cestu pro CRL (i případné přírůstkové CRL) až k důvěryhodné kotvě.
20. Verifikujeme certifikáty v certifikační cestě CRL (i případného přírůstkového CRL) (viz předchozí odstavec).
21. Ověříme elektronický podpis CRL (i přírůstkového CRL).
22. Prohledáme přírůstkové CRL není-li na něm uveden ověřovaný certifikát.
23. V případě, že ověřovaný certifikát nebyl uveden na přírůstkovém CRL (nebo se přírůstkové CRL vůbec nepoužívá), pak prohledáme úplné CRL.
24. Nebyl-li certifikát nalezen ani na úplném CRL, pak jej považujeme za neodvolaný (pomocí mechanismu CRL). Časové razítko

8 Časová razítka

Certifikát veřejného klíče je datová struktura, která spojuje identifikaci žadatele s jeho veřejným klíčem. Tato vazba je stvrzena elektronickým podpisem nezávislé třetí strany, kterou nazýváme certifikační autorita. Časové razítko je svým způsobem obdobná datová struktura, která však svazuje obsah zprávy (dokument) s určitým časem.

Časové razítko je totiž datová struktura, která mj. obsahuje čas, otisk z dokumentu, vydavatele razítka a pořadové číslo. To vše je stvrzeno nezávislou třetí stranou – Autoritou pro vydávání časových razítek (*Time stamping authority* – TSA). Časové razítko tak slouží jako důkaz, že dokument (přesněji řečeno otisk z dokumentu) existoval v daném čase. Dokument je v časovém razítku reprezentován pomocí tzv. *message imprint*, což je dvojice: otisk z dokumentu a algoritmus pro výpočet otisku, kterým byl otisk z dokumentu spočten.

Jelikož TSA nezkoumá totožnost žadatele a podle svého standardu tak činit ani nesmí, neobsahuje časové razítko identifikaci žadatele. Časové razítko tudíž není důkazem o tom, že nějaký dokument měla v okamžiku vydání časového razítka v držení konkrétní osoba, ale jen indicií, že dokument existoval v konkrétním čase.

Podívejme se, k čemu můžeme časové razítko využít v praxi.

Příklad 1 – razítkování dokumentů

Aplikace, která každého ihned napadne, je razítkování dokumentů: k existujícímu dokumentu vytvoříme časové razítko a vznikne dvojice dokument + časové razítko. Časové razítko nám nejenom slouží jako indicie o existenci dokumentu v čase, ale i zapouzdřuje dokument v čase. Jakákoliv pozdější změna dokumentu by totiž způsobila neplatnost časového razítka.

Většina TSA dává k dispozici software, kterým si můžeme k souborům vytvářet časová razítka, která pak ukládáme v samostatných souborech (zpravidla s příponou `.tsr`). Praktický význam razítkování samotných dokumentů je však omezený.

Příklad 2 – razítkování auditních záznamů

Razítkování auditních záznamů („logů“) je jednou z komplikovanějších aplikací předchozího příkladu.

Pokud se hacker dostane do počítače a něco zde provede, snaží se po sobě zamést stopy tím, že opraví auditní záznamy. Pravidelným razítkováním předchozích auditních záznamů, znemožníme útočníkovi změnit záznamy bez povšimnutí. Jedinou možností pro útočníka pak je odstranit i časová razítka, čímž ovšem po sobě také zanechá stopy tím, že je porušena perioda vkládaných auditních záznamů s časovým razítkem.

Příklad 3 – rozšířený elektronický podpis

Odešleme-li do banky digitálně podepsaný platební příkaz, který banka archivuje např. 10 let, pak po vypršení platnosti certifikátu, který byl použit k podpisu (např. po roce), dochází k problematické situaci. Odesilatel může prohlásit, že takový příkaz neodeslal on. Proč? Může totiž tvrdit, že banka měla k dispozici jeho veřejný klíč po dostatečně dlouhou dobu a po tuto dobu k němu hledala příslušný soukromý klíč, kterým posléze sama podepsala falešný platební příkaz. Pokud si ovšem banka před archivací platebního příkazu nechá vytvořit časové razítko z elektronického podpisu platebního příkazu, má důkaz, že byl platební příkaz proveden skutečně v době platnosti uživatelova certifikátu a tedy uživatelem samotným.

Z tohoto smyšleného příkladu nám vyplývá několik závěrů pro digitálně podepsané dokumenty:

- Časové razítko z elektronického podpisu je mnohdy významnějším důkazem než časové razítko ze samotného dokumentu. Dokument totiž mohl být vytvořen podstatně dříve, než byl podepsán a významné je, kdy byl digitálně podepsán a nikoliv kdy byl vytvořen.
- Je možné, že se bude jedna ze stran chtít zpochybnit platnost dokumentu, který byl podepsán oběma stranami. Po podpisu dokumentu proto nahlásí ztrátu privátního klíče (odvolá certifikát). Bez existence časového razítka daného dokumentu již nelze dokázat, zda byl dokument podepsán před odvoláním příslušného certifikátu nebo až po něm.

- V případě dokumentů s relativně dlouhou dobou platnosti je situace ještě horší. Elektronický podpis se verifikuje pomocí příslušného certifikátu. Ten je ale platný v intervalu notBefore po notAfter. Pokud elektronický podpis nikterak neošetříme, je platnost elektronickýho podpisu po uplynutí tohoto intervalu sporná. Může se využívat zmíněný argument, že útočník měl k dispozici veřejný klíč tak dlouhou dobu, aby byl technicky schopen nalézt k němu příslušný soukromý klíč. Časové razítko z elektronickýho podpisu je pak důkazem, že dokument byl vytvořen v době, kdy útočník neměl k dispozici dopočítaný soukromý klíč, takže podpis dokumentu je pravý.

Obnovováním elektronických podpisů se budeme věnovat v kap. 26 Dlouhodobý elektronický podpis. Dlouhodobou archivací digitálně podepsaných dokumentů se pak budeme věnovat v kap. 27 Dlouhodobá archivace nejenom digitálně podepsaných dokumentů.

8.1.1 Co to je čas?

Časové razítko obsahuje čas. Otázkou je jaký čas máme na mysli a jak je přesný. Stranou by neměla zůstat ani otázka, zdali TSA umí doložit věrohodnost času, který udává ve svých časových razítkách.

Čas události je abstraktní hodnota určující pořadí události mezi ostatními událostmi v určitém časovém rozmezí. Čas se měří na základě periodicky se opakujících přírodních jevů - oscilátorů. Oscilátor je generátor poskytující v rámci dané tolerance přesnou frekvenci. Všeobecně známé jsou astronomické oscilátory založené na pohybu těles ve Sluneční soustavě (rotace Země kolem osy, otáčení Země kolem Slunce nebo otáčení Měsíce kolem Země). Jinými typy oscilátorů jsou: kyvadlo hodin či krystalové nebo atomové oscilátory. Hodiny pak tvoří oscilátor spolu s čítačem, který počítá počet cyklů. Např. jedno otočení Země kolem své osy je jeden den. Jedna perioda kyvu kyvadla pak může trvat např. 1,3 s.

Čas vyjadřujeme v kalendářních dnech, hodinách, minutách a sekundách a jejich zlomcích. A jelikož čas odvozujeme od rotace Země, musíme ještě uvést časové pásmo.

V průběhu historie se měnil pohled jak na kalendář, tak i na specifikaci délky sekundy.

8.1.1.1 Kalendář

Na základě rotace Země určujeme den; na základě oběhu Země kolem Slunce máme solární kalendář; na základě oběhu Měsíce kolem Země pak lunární kalendář. Vzájemná nesouměřitelnost těchto kalendářů vede k různým úpravám kalendářů.

Římský republikánský kalendář má základ v lunárním roce. V r. 50 před Kristem se tento kalendář rozcházel se slunečním rokem o osm týdnů. Sám Julius César pozval alexandrijského astronoma Sosigenese aby odstranil tento nesoulad. A tak v r. 46 před Kristem vznikl tzv. Juliánský kalendář., který je založen na 365 denním roce s tím, že každý čtvrtý rok se navíc ještě vkládá jeden přestupný den. Prvních 36 let se však vkládal přestupný den ne každé čtyři roky ale již po třech letech. Výsledkem bylo vložení 12 místo správných 9 přestupných dní. V počátcích fungování Juliánského kalendáře – následovala řada korekcí, ale tato chyba byla zcela opravena až v r. 8 našeho letopočtu.

Roku 1582 vydal papež Řehoř XIII papežskou bulu, v níž kromě jiných věcí také stanovil délku trvání solárního roku na 365.2422 dnů. Výsledkem reformy papeže Řehoře XIII bylo, že po 4. říjnu 1582 následoval 15. říjen 1582. Tím se odstranily chybně vložené dny do juliánského kalendáře a rovněž se provedla korekce, že rok je o něco kratší než 365,25 dne. Aby se dosáhlo co nejpřesnějšího výsledku, začal se vypouštět přestupný den v roce dělitelném 100, který ale není dělitelný 400. Tím bylo docíleno hodnoty 365,2425, tj. rozdílu asi 26 vteřin oproti naměřeným hodnotám. Vznikl tak Gregoriánský kalendář.

Před rokem 1900 bylo do Juliánského kalendáře vloženo 474 přestupných dní. Gregoriánský kalendář z nich vypustil 14 dnů. Gregoriánský kalendář je dnes všeobecně rozšířen, ale ještě mnohé země začátkem 21. století využívaly starý Juliánský kalendář, takže se klidně stalo, že v Rusku měli říjnovou revoluci (1917) v listopadu.

8.1.1.2 Délka dne a sekunda

Čas definujeme na základě tabulek hodnot, které určují některé časové intervaly jako např. tropický rok, který je definován jako jedna otočka Země okolo Slunce. Tato hodnota se stanovuje na základě pozorování Slunce, Měsíce a dalších planet.

Roku 1958 byla standardní sekunda stanovena na 1/31,556,925.9747 tropického roku na počátku 20. století. Což znamená, že tropický rok má 365.2421987 dnů. Protože však ve vesmíru není nic stálého, přesnost, se kterou jsme schopni určit tropický rok, je cca. 50 ms a každý rok přičítáme k jeho délce asi 5.3 ms.

Země a ostatní astronomická tělesa jsou ohromná tělesa – ohromné setrvačníky, takže by se dala oče-

kávat vysoká stabilita těchto oscilátorů. Avšak moderní elektronika využívá podstatně stabilnější oscilátory. Roku 1967 se standard definující vteřinu změnil na „dobu trvání 9,192,631,770 period záření, které přísluší přechodu mezi dvěma hladinami velmi jemné struktury základního stavu ^{133}Cs“.

Od 1972 jsou světové standardy definující čas a frekvenci založeny na Mezinárodním atomovém Čase - *International Atomic Time* (TAI), který určují a udržují sady cesiových oscilátorů s přesností větší než jedna mikrosekunda za den. Musíme si ale uvědomit, že sluneční čas, kterým se řídíme, nemusí přesně odpovídat času atomovému.

8.1.1.3 Přestupné vteřiny, UTC

Den se dělí na 24 hodin, 1440 minut či 86400 vteřin. Pokud základní jednotkou je vteřina, pak by i každý den měl být tvořen 86400 vteřinami. Jenže problém je v tom, astronomická měření tomu neodpovídají. Je tedy třeba koordinovat čas astronomický s časem atomovým.

Přesný atomový čas se upraví podle nepřesného astronomického času. Je to tak proto, že podle toho nepřesného astronomického času běží naše biologické rytmy. A čas se přece pije o páté podle našich vnitřních biologických hodin.

Korekce se provádí, když se časová odchylka přiblíží k 0,7 vteřiny, tehdy se do TAI časového rozpětí vkládá/odebírá tzv. přestupná vteřina. Přestupná vteřina se vkládá/odebírá nikoliv libovolně, ale vždy jen v posledním dne června nebo prosince. Přestupná vteřina se vkládá za vteřinu 23:59:59 jako vteřina 23:59:60. Při vypouštění se vypouští vteřina 23:59:59. První přestupná vteřina byla vložena 1. ledna 1972. Výsledkem je tak koordinovaný čas (*Universal Time Coordinated*) ,který se označuje jako UTC.

8.1.1.4 Časové zóny, letní čas

Zatím jsme se na čas hleděli očima pozorovatele v Greenwich v Londýně. Jelikož je čas odvozen od rotace Země, musíme určit, kde na Zemi se pozorovatel nachází. Vznikají tak časová pásma. Povrch země je administrativně rozdělen do 24 základních časových pásem postupně posouvaných po jedné hodině. Některým zemím posun o celé hodiny nevyhovoval, tak existují i země s časovým pásmem posunutým o ¼, ½ či ¾ hodiny vůči okolním zemím.

Časové pásmo vyjadřujeme jeho posunem vůči časovému pásmu, ve kterém leží Greenwich. Směrem na východ se čas přičítá (časová pásma mají kladné znaménko); směrem na západ se čas odečítá (časová pásma mají záporné znaménko).

Greenwich leží v nultém časovém pásmu. Toto časové pásmo se též označuje GMT (*Greenwich Mean Time*). V tomto případě namísto časového pásma je v časovém údaji písmeno Z jako nulové (*zero*) časové pásmo, tj. časové pásmo ve kterém leží observatoř Greenwich. V angloamerické literatuře (zejména populární) s GMT označuje jako „Zulu“. Slovo "Zulu" má kořeny v anglickém radiovém provozu, kdy se písmeno Z hláskuje jako „Zulu“ (*„Z“ for Zulu*). Např. místní čas v Kátmandu je posunut o 5 hodin a 45 minut na východ od Greenwich.

Problematika časových pásem je navíc komplikována využíváním letního času. Praktickou poučkou tedy je čas v datových strukturách uvádět v GMT a místní čas uvádět jen při zobrazování časových údajů uživateli. Např. certifikáty i časová razítka zásadně obsahují GMT.

8.1.1.5 Počítačový čas

Čas určuje oscilátor spojený s čítačem, který zaznamenává počet cyklů. Mnoho počítačových protokolů včetně systémového času mnohých operačních systémů se vnitřně nepoužívá ve tvaru kalendářního dne, hodin, minuty, vteřin a časové zóny, ale má jednouchý čítač, který se neustále navyšuje. Aplikace si pak samy provádí transformaci hodnoty tohoto čítače do sice složitého tvaru, ale pro člověka dobře čitelného tvaru.

Např. protokol NTP používá čítač representovaný 64 bitovým celým číslem bez znaménka, který obsahuje vteřiny od 00:00:00 dne 1. ledna 1900. Toto 64 bitové číslo je rozděleno na dvě poloviny po 32 bitech. První polovina obsahuje vteřiny a druhá polovina pak desetiny (zlomky) vteřin (jako by byla desetinná čárka uprostřed tohoto 64 bitového čísla).

Do 32 bitů se vejde 136 let v sekundách. Tj. v roce 2036 tento čítač přeteče – budeme tedy řešit problém roku 2036. Musíme si uvědomit, že přepočet čítače na kalendářní datum není až tak jednoduché – nesmíme zapomenout na přestupné vteřiny. Např. první přestupná vteřina z 1.1.1972 měla hodnotu NTP čítače 2,272,060,800.

Unix používá rovněž 32 bitový čítač avšak se znaménkem, které zabírá jeden bit, takže Unixový čítač přeteče již po 68 letech. Unix však nepočítá vteřiny od roku 1900, ale od 1.1.1970. Takže přeteče až začátkem roku 2038. Nedávno (11. ledna 2004 00:37:04 GMT) tak UNIX oslavoval své kulaté 0x40000000 narozeniny (ve vteřinách)! Není to už žádný mladík, přehoupl se do své druhé poloviny.

Protokol *„ICMP timestamp“* běží v milisekundách od předchozí půlnoci. Atd.

8.1.1.6 Zdroje času

Primárním zdrojem času musí být extrémně přesný oscilátor, jako je např. již zmíněný přechod mezi dvěma stavy atomu. Dostupné oscilátory jsou založeny na přechodu atomů vodíku, cesia a rubidia. Zmíněné oscilátory mají stabilitu:

- Vodík – 2.10^{-14} sekundy za den,
- Cesium - 3.10^{-13} sekundy za den,
- Rubidium - 5.10^{-12} sekundy za den.

Pro srovnání krystalový oscilátor má stabilitu podle provedení. Vysokou stabilitu 10^{-9} sekundy za den má teplotně řízený krystal („v peci“). Kdežto běžný teplotně neřízený krystal má stabilitu 10^{-6} sekundy za den, ale na každý teplotní stupeň.

Neméně důležitým parametrem je cena zdroje času. Je vcelku pochopitelné, že národní laboratoř zabývající se časem si zaplatí nejkvalitnější zdroje času, kdežto na PC nám bude stačit běžný teplotně neřízený krystal. Např. autorita pro vydávání časových razítek (TSA) musí být vybavena interním zdrojem času přesností odpovídajícím její politice pro vydávání časových razítek. Komerční TSA zpravidla využívají rubídiové oscilátory.

8.1.1.7 Poskytovatelé času

Zdroje času nemohou být dva. Pokud by se totiž rozešly, nedalo by se zjistit, které s nich jdou správně a které špatně. Laboratoří zabývajících se měřením času je na světě celá síť. Tyto laboratoře si vzájemně koordinují čas. Výsledný čas je tedy jakousi střední hodnotou, do které jsou zahrnuty časy jednotlivých laboratoří. Avšak ne každá laboratoř má stejnou váhu pro svůj čas. Laboratoře se stabilnějším časem mají větší váhu než laboratoře, jejichž čas se v minulosti více odchyloval od stanoveného času.

Nyní již víme, že existuje celosvětová síť laboratoří vybavených kvalitními oscilátory, které jsou navíc celosvětově koordinovány. Tyto laboratoře mají k dispozici z pohledu nás uživatelů přesný čas. Nyní zbývá vyřešit problém, jak dopravit přesný čas až k nám - uživatelům.

Mnohé laboratoře zabývající se časem poskytují svůj čas veřejnosti. Čas je poskytován různými způsoby, např.:

- Přes běžné rozhlasové/televizní vysílání. Tato informace je určena pro člověka. Při pohledu na hodiny zobrazené na TV obrazovce si pak seřizuji své kapesní hodinky či hodiny v mobilu.
- Je vysílán na speciálních rozhlasových frekvencích (od dlouhých až na velmi krátké). Informace z těchto frekvencí je přímo počítačově zpracovatelná. Používá se termín radiové hodiny. Rozhlasový přijímač může být přímo napojen na počítač a podle rozhlasového signálu se tak přímo může koordinovat zdroj času počítače. Radiové hodiny mají bohatou historii a prvotně nebyly určen pro synchronizaci času počítačů, ale jako radionavigační systém. Veřejnost zná radiové hodiny v souvislosti s automatizovaným seřizováním elektronických hodin. Od nádražních hodin až po domácí budíky. TSA však potřebuje stabilnější čas než budík nebo nádražní hodiny….
- GPS je družicový navigační systém, který rovněž poskytuje čas. GPS je přímo určen pro počítačové zpracování; je výrazně přesnější než rozhlasový čas. GPS postupně vytlačuje používání rozhlasového čas při synchronizaci času počítačů.
- Přes počítačovou síť aplikačními protokoly (např. NTP).
- Poskytují tzv. zaručený čas, který však již nebývá zadarmo.

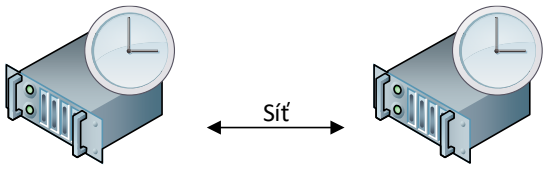
8.1.1.8 Synchronizace času přes síť

Představme si, že Alois nějakou cestou získává přesný čas. Nyní pro nás není důležité, jak k němu přišel, ale že systémové hodiny jeho počítače mají přesný čas. Bohumila by chtěla získávat od Aloise přesný čas a k tomuto účelu bych chtěla využít počítačovou síť, např. na bázi protokolu TCP/IP ().

Problém je v tom, že když Alois zapíše přesný čas do paketu protokolu TCP/IP pošle jej přes Internet, bude tento paket bloudit Internetem, až dorazí k Bohumile. Nejhorší na tom je, že doba bloudění Internetem bude pokaždé jiná. Jestliže Alois odešle Bohumile další paket, bude doba jeho bloudění s největší pravděpodobností jiná než doba bloudění předchozího paketu.

Bohumile je jasné, že své hodiny nemůže nastavit na hodnotu uvedenou v obdrženém paketu, ale musí jejich nastavení posunout o dobu bloudění paketu Internetem. Jenže ona neví, jak dlouho paket Internetem bloudil, protože nemá přesný čas. Ten chce přece teprve získat!

Musíme se tedy na to podívat z jiného úhlu. Entomolog, když potřebuje popsat nový druh brouka, zjišťuje délku brouků tohoto druhu. Jak to udělá, když každý brouk je trochu jiný? Prostě vzpomene si na přednášku z matematické statistiky, kterou kdysi absolvoval při svém studiu a hned ví jak na to. Je mu jasné, že musí nachytat dostatečné množství brouků a statisticky jej vyhodnotit a pak může např. tvrdit, že s 95% pravděpodobností brouci tohoto druhu jsou dlouzí jeden palec.



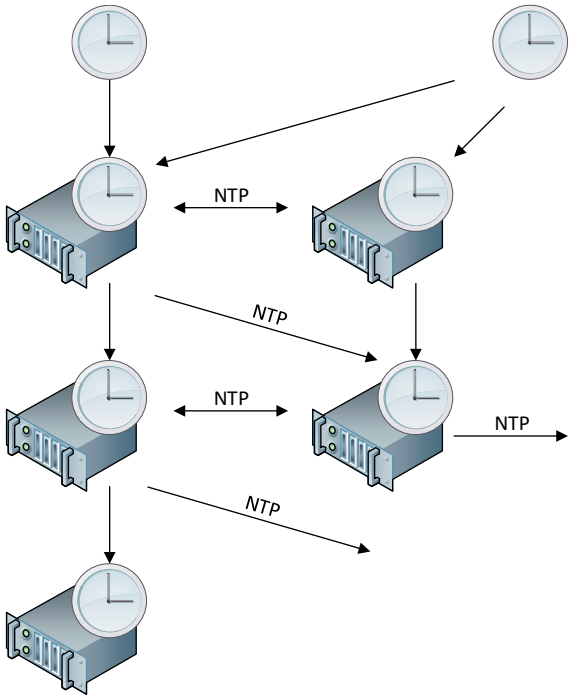
Obrázek 8.2 JObrázek 8.2ak seřídit čas přes síť

Takže těch paketů se prostě pošle dostatečné množství, aby Bohumila z tohoto množství obdržených Aloisových paketů byla schopna dostatečně přesně nastavit čas na svých systémových hodinách. Jedním z konkrétních řešení jak to prakticky řešit je Network Time Protocol – NTP (RFC 5905).

Aby Alois s Bohumilou nemuseli komunikovat ručně, spustí na svých strojích procesy a ty to vzájemně zajistí aplikačním protokolem NTP. NTP je aplikační protokol, který nevyužívají uživatelé, ale používají jej systémy, aby si automatizovaně nastavovaly čas.

Jenže v Internetu nejsou jen Alois a Bohumila. Existují servery, které mají k dispozici přesný čas z nějakých primárních zdrojů (národní referenční laboratoře, GPS apod.). Tyto servery se označují jako Stratum 1 (Obrázek 8.1). Servery Stratum 1 si koordinují čas vzájemně a navíce mohou poskytovat čas dalším systémům, které už nejsou přímo napojeny na primární zdroje času. Tyto systémy (*hosts*) se označují jako stratum 2. Obdobně jsou systémy stratum 3 atd.

Protokol NTP je jedním z mála Internetových aplikačních protokolů, které striktně nedodržují architekturu klient/server.



Obrázek 8.1 NTP servery

Pokud stojíme před problémem nastavit čas na našem systému, zásadou je, že se budeme snažit získávat čas z alespoň tří na sobě nezávislých zdrojů času. Každý zdroj času je obsluhován pouze lidmi a může se tudíž rozejít. Pokud bychom brali čas pouze ze dvou zdrojů a jeden se rozešel, nepoznáme, který je ten správný. Rovněž útočníkům útočícím na čas tím ztěžujeme jejich práci.

8.1.1.9 Zaručený čas

Léta používáme čas šířený z Internetových NTP serverů. Využíváme UDP datagramy a nepoužíváme žádné zabezpečení – např. autentizace zdroje času (pokud za autentizaci nepovažujeme IP-adresu zdroje času).

V poslední době se však objevily bezpečnostní požadavky i na nastavení času. Tyto požadavky jsou dvojího druhu:

- Aby komunikace při nastavování času byla bezpečná, tj. aby tuto komunikaci nemohl zneužít útočník a způsobit tak špatné nastavení času.
- Aby po nastavení času systému systém obdržel nějaký důkaz (nějakou certifikaci), že má správně nastavený čas. Např. když TSA vydá časové razítko, aby k němu mohla přidat certifikát o tom, že její čas je tak a tak přesný. A kdyby takový certifikát vydávala národní referenční laboratoř pro čas, pak by to už byl i úřední doklad s právní váhou. Takže k elektronickému podpisu bych pak přidával nejenom časové razítko z elektronického podpisu, ale ještě certifikát o tom, že TSA vyplňuje tak a tak přesný čas. Příkladem takového certifikátu je TAC (Time Attributte Certificate). TAC je atributový certifikát (vydávaný např. americkým NISTem*) stvrzujícím tím úspěšně provedenou kalibraci hodin). Tento atributový certifikát má pak platnost např. 14 dnů, po kterých

*) NIST = National Institute of Standards and Technology (USA)

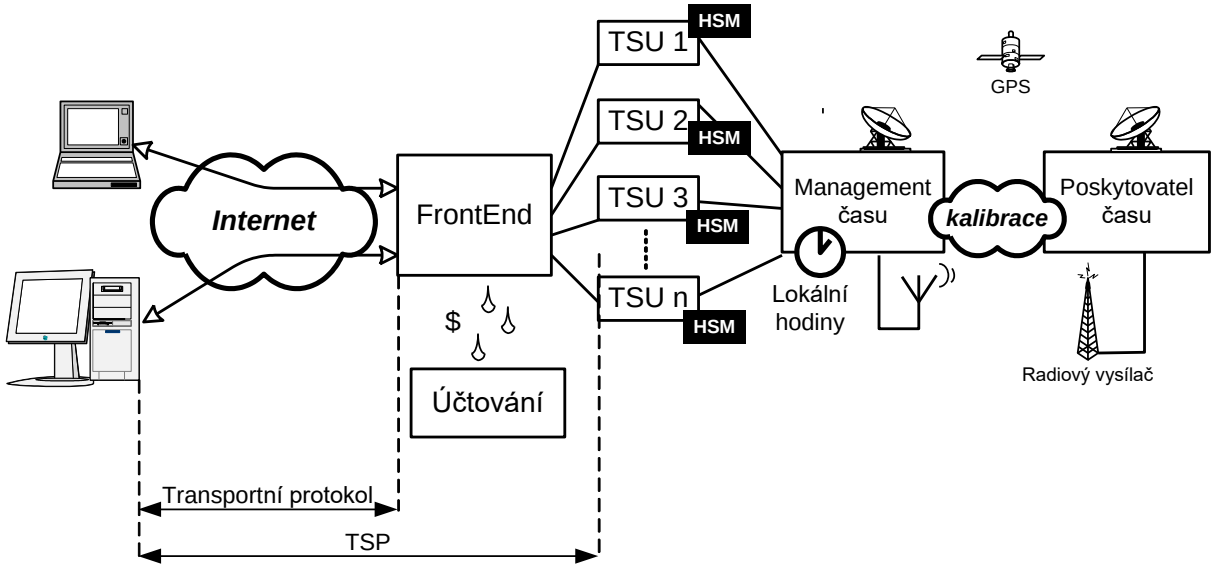
mohou kalibrované lokální hodiny garantovat dostatečnou stabilitu času.

Bohužel dosud neexistuje žádný všeobecně platný mezinárodní standard, který by řešil tuto problematiku (ani TAC není podložen žádným standardem). Výsledkem je, že se tato problematika řeší na úrovni jednotlivých dodavatelů. Velice zajímavým řešením je např. řešení TSA firmy nCipher. Sama TSA je řešena jako aplikace běžící přímo v HSM modulu doplněném o dostatečně stabilní hodiny (např. s rubidiovým oscilátorem). Je pak na provozovateli TSA, jestli si kalibraci těchto hodin sjedná s NISTem nebo např. využije NTP servery v Internetu.

8.2 TSA

Na obrázku Obrázek 8.3je znázorněna obvyklá architektura autority pro vydávání časových razítek. Klient přistupuje transportním protokolem přes Internet na FrontEnd TSA, který požadavek předá na vyřízení konkrétní TSU (*Time Stamp Unit*). Vlastní TSA se totiž zpravidla skládá z několika vydávacích jednotek TSU. TSU je množinou hardware a software určená pro vydávání a zejména podepisování časových razítek pro konkrétní politikou TSA. Každá TSU má vlastní pár klíčů; přičemž soukromý klíč zpravidla bývá uložen v HSM[*)].

Klient žádá o vydání časového razítka pomocí datové struktury nazývané žádost o časové razítko. Tuto žádost zašle na FrontEnd TSA. Ten pak podle politiky vydávání časových razítek předá žádost konkrétní TSU. TSU vrátí odpověď, která, pokud nedošlo k nějaké formální chybě, obsahuje požadované časové razítko. Tato komunikace je specifikována pomocí protokolu pro vydávání časových razítek (*Time Stamping Protokol* –TSP), který rovněž specifikuje formát žádosti i formát časového razítka (Obrázek 8.5).



Obrázek 8.3 Architektura autority pro vydávání časových razítek

Zatímco při synchronizaci za využití NTP si na Internetu najdeme více NTP serverů (minimálně tři) a pomocí nich si synchronizujeme svůj čas, v případě zaručeného času (např. při kalibraci na NIST) se vytváří bezpečný kanál obdobně jako v TLS mezi naším systémem a systémem poskytovatele. Obě strany se autentizují: poskytovatelův systém proto, abychom my si byli jisti, že nekomunikujeme se systémem hackera a uživatel (kalibrované hodiny) se autentizuje poskytovateli, aby mj. věděl, že čas poskytuje platícímu zákazníkovi. Jelikož je zdroj času důvěryhodný a komunikační kanál bezpečný, stačí nám jeden zdroj času.

TSP se nezabývá specifikací přenosu dat mezi klientem a TSA. Takže žádost o časové razítko může být uložena i na disketu a odnesena na TSA. V případě síťové komunikace TSP vkládá žádost i odpověď do odpovídajícího transportního protokolu (např. HTTP). K oběma případům se ještě vrátíme.

[*)] HSM moduly podporují i více párů klíčů a mohou být sdíleny mezí více TSU skrze vyhrazenou síť.

Dosud jsme představili TSA jako šíleného úředníka, který cokoliv co mu přijde pod ruku, pouze mechanicky časově orazítkuje. Mezi jedenácti základními přikázáními TSA, jak je definuje RFC-3161, totiž je: „(4) TSA vydá časová razítka vždy, když je to možné po obdržení platné žádosti o časové razítko.“ Avšak to je jen teorie, v praxi TSA musí z něčeho žít (musí vydělávat). Takže v praxi bude TSA vydávat časová razítka jen platícím klientům, tzn., že klient se bude muset vůči TSA nějakým způsobem autentizovat – např. osobním certifikátem v SSL/TLS komunikaci (tj. TSP je např. vložen do HTTPS kanálu).

Autentizace klienta pomoci jeho uživatelského certifikátu se jeví jako docela praktická. Proč? Není to jen z hlediska bezpečnosti, ale obvykle TSA provozuje nějaká certifikační autorita (jako organizace). Pak certifikáty této CA lze navíc využít pro autentizaci vůči TSA. Konec konců, když už klient má osobní certifikát s tím, že je založen v databázi CA, pak není nic jednoduššího než tento certifikát použít i k autentizaci vůči TSA.

Velice důležité je, aby TSA používala stabilní zdroj času. K tomuto účelu slouží management času, který využívá čas od důvěryhodných poskytovatelů času. V případě menších TSA se bere čas z jednoho zaručeného zdroje času. V případě větších TSA je praktické brát čas alespoň ze tří zdrojů (ze dvou to nemá smysl, což jsme si již vysvětlili – v případě tří zdrojů už je možné hlasování).

TSA je důvěryhodná třetí strana. Provozovatel TSA vytvoří jeden nebo více dokumentů nazývaných politika vydávání časových razítek (politika TSA). Zpravidla se vytváří pro každou TSU jedna politika. Každé politice se pak přiřadí jedinečný identifikátor objektu (OID). Identifikátor objektu politika TSA, pod kterou bylo časové razítko vydáno, se vždy vkládá do časového razítka. (To je rozdíl oproti certifikátu, který rozšíření Certifikační politiky může a nemusí obsahovat a toto rozšíření může obsahovat i více politik!)

TSA musí podepisovat časová razítka klíčem výhradně určeným k tomuto účelu. TSA může mít více párů klíčů (pro každou TSU jeden), např. pro různé politiky, různé algoritmy, různé délky klíčů apod. Veřejný klíč TSA je uložen v certifikátu TSA, který musí mít právě jednu instanci rozšíření „Rozšířené použití klíče“, které je označeno jako závažné.

Před tím než započnete s výstavbou TSA a zejména před tím než začnete psát politiky TSA, prostudujte si RFC-3628: *Policy Requirements for Time-Stamping Authorities (TSAs)*. A nikdy nezapomeňte na 11 základních přikázání RFC-3161:

25. Použijte důvěryhodný zdroj času (důvěryhodnost zdroje času by měla být blíže specifikována v politice TSA).
26. Vkládejte důvěryhodný čas do každého vydaného časového razítka.
27. Do vydávaných časových razítek vkládejte jedinečné pořadové číslo. Tj. každé vydané časové razítko musí mí své jedinečné pořadové číslo (v rámci TSA).
28. TSA vydá časová razítka vždy, když je to možné po obdržení platné žádosti o časové razítko (princip -„šílený úředník“).
29. Do každého časového razítka vkládejte identifikátor politiky TSA, pod kterou bylo razítko vydáno.
30. Algoritmus pro výpočet otisku musí být jednoznačně identifikován svým identifikátorem objektu (OID).
31. Zkontrolujte, zdali délka otisku odpovídá použitému algoritmu pro výpočet otisku (specifikovaném příslušným identifikátorem objektu (OID)).
32. Nesnažte se žádným způsobem analyzovat otisk jinak, než že zkontrolujete jeho délku. (Např. nesnažte se k otisku hledat původní dokument.)
33. Do časového razítka nevkládejte žádnou identifikaci žadatele o časové razítko.
34. K podepisování časových razítek využívejte jen klíče výhradně určené k tomuto účelu a odpovídající certifikátu TSA.
35. Jestliže žadatel použil v žádosti o časové razítko rozšíření, které TSA nepodporuje, TSA vrátí místo časového razítka označení chyby (nevydá časové razítko).

Velice důležitou součástí TSA je její dokumentace, která je v mnohém podobná dokumentaci certifikační autority.

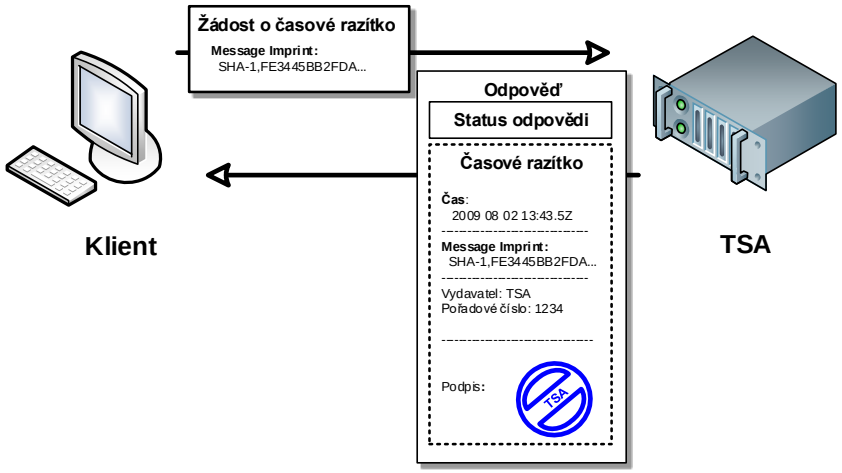
8.3 Protokol pro vydávání časových razítek (TSP)

Protokol pro vydávání časových razítek (TSP) je specifikován RFC-3161. Protokol TSP se skládá z žádosti o časové razítko a z odpovědi na tuto žádost.

Komunikace je jednoduchá: klient pošle žádost, na kterou TSA vrací odpověď. Odpověď buď obsahuje vydané časové razítko, nebo specifikaci chyby. Po vložení protokolu TSP do transportního protokolu (např. HTTP) vznikne protokol typu klient-server.

Důležité je, že žádost o časové razítko není digitálně podepsaná, tj. je anonymní. TSA nezkoumá totožnost žadatele. TSA vydá časové razítko vždy, když žádost o časové razítko je formálně správně. Naproti tomu časové razítko je digitálně podepsaná struk-

tura, která mj. obsahuje identifikaci TSA a pochopitelně také otisk zkopírovaný z žádosti o časové razítko.



Obrázek 8.4 Protokol pro vydávání časových razítek (TSP)

Hypoteticky je možné časové razítka vydávat i dávkově (*off line*). Kdy klient zašle žádost na TSA a ta nevydá časové razítko bezprostředně, ale někdy později. Na první pohled je toto řešení vhodné pro vydávání časových razítek přes e-mail. Může však mít i své opodstatnění v případě, že požadovaný TSU je v krátkodobé poruše nebo přetížena a klient nemůže tak dlouho čekat na odpověď TSA.

Dávkové vydávání časových razítek může být zajímavé zejména v případě, když např. banka přijímá ohromné množství podepsaných platebních příkazů přes Internet. Časové razítko se v takovém případě vkládá do elektronického podpisu pro jeho ošetření, tj. není důležitý přesný okamžik vydání razítka, ale musí být vydáno v dostatečně krátké době, než certifikát klienta vyprší nebo bude odvolán. Tento proces může probíhat např. přes noc, kdy banka není přetížena.

8.3.1 Transportní protokoly

Pokud se žádost neukládá na datový nosič a osobně se neodnese na TSA, pak se TSP komunikace vkládá do transportních protokolů, které zajistí přenos dat mezi klientem a TSA (tj. serverem).

Jako transportní protokoly je možné použít přímo protokoly TCP nebo teoreticky i UDP. Pro tyto účely jsou pro TSA vyhrazeny dobře známé porty 318/tcp a 318/udp. Běžnější je však vložit tuto komunikaci do protokolu HTTP nebo elektronické pošty. Elektronická pošta je pak obdobou dávkového zpracování.

8.4 Žádost o časové razítko

Žádost o časové razítko je jednoduchá sekvence:

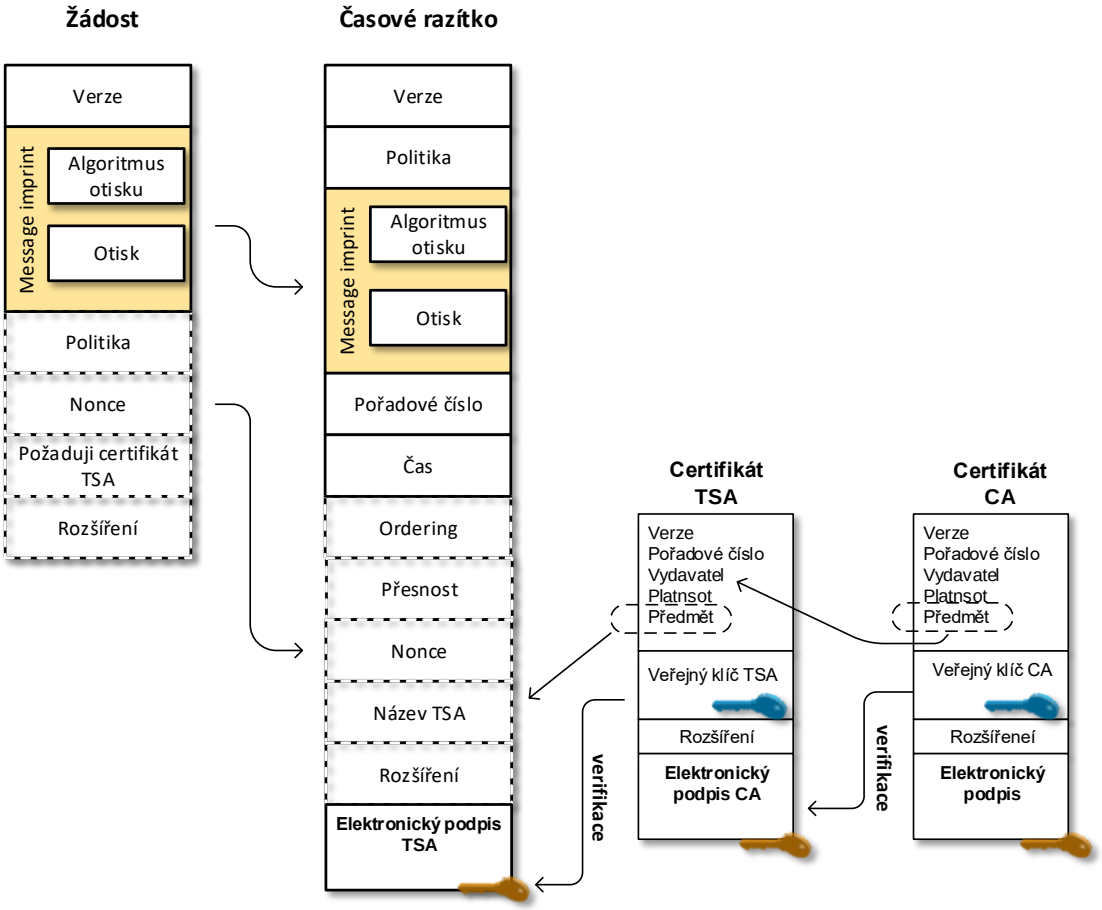
Význam jednotlivých položek:

- Položka Verze obsahuje verzi protokolu TSP. Aktuální verze je 1.
- Položka Message imprint obsahuje otisk razítkovaného dokumentu a algoritmus, kterým byl vytvořen.
- Volitelná položka Politka může obsahovat identifikátor objektu politiky, pod kterou si klient přeje vydat časové razítko.
- Volitelná položka Nonce obsahuje dostatečně velké náhodné číslo, které TSA kopíruje se do odpovědi.
- Pokud je položka „Požaduji certifikát TSA“ nastavena na hodnotu TRUE, žadatel si přeje, aby TSA spolu s časovým razítkem ve své odpovědi též vrátila certifikát TSA (certifikát pro ověření elektronického podpisu časového razítka). Položka Rozšíření je obdobou rozšíření certifikátu.

8.4.1 Časové razítko

Časové razítko je elektronicky podepsaná datová struktura (formát CMS), která obsahuje:

- Položka Verze obsahuje verzi protokolu TSP. Současná verze je 1.
- Položka Politika obsahuje identifikátor objektu politiky TSA, pod kterou bylo časové razítko vydáno a může být využito.
- Položka Message imprint obsahuje otisk razítkovaného dokumentu a algoritmus, kterým byl vytvořen (zkopírováno z žádosti).
- Položka Pořadové číslo obsahuje pořadové číslo vydaného časového razítka. Logicky bychom předpokládali, že se sériová čísla vydávaných razítek budou zvyšovat např. o 1. Ale není tomu tak, podle jaké logiky budou sérová čísla přidělována nestanoví žádný předpis – ten říká pouze, že v rámci konkrétní TSA musí být vydaná sériová čísla časových razítek jedinečná. Díky tomu nám bohužel sériová čísla časových razítek neříkají nic o pořadí jejich vydávání. V politice TSA však může být stanoven způsob přidělování pořadových čísel konkrétní TSA.



Obrázek 8.5 Žádost o časové razítko a časové razítko

- Položka Čas obsahuje čas vydání razítka. Používá se UTC čas a nevyužívají se lokální časová pásma, tj. čas končí písmenem „Z“ používaným pro GMT.
- Položka Přesnost vyjadřuje přesnost času. Maximální odchylka může být uvedena v sekundách, milisekundách a mikrosekundách. Pokud některá položka není uvedena, bere se jako by měla nulovou hodnotu.
- Položka Ordering specifikuje, zdali je čas uváděný na časových razítkách tak přesný, že pomocí něj mohou být vydaná časová razítka seřazena tak, jak byla vydávána. Jestliže je toto pole v časovém razítku přítomno a má hodnotu TRUE, pak časová razítka mohou být seřazena tak jak byla vydána podle času v nich uvedených.
- Volitelná položka Nonce se vyplní obsahem stejnojmenné položky z žádosti o časové razítko (byla-li vyplněna). Tato položka pak slouží k párování vydaného časového razítka s příslušnou žádostí o časové razítko.
- Volitelná položka Název TSA může obsahovat jméno autority TSA. Překvapivě je tato položka volitelná, protože název TSA lze vzít z předmětu certifikátu, kterým se verifikuje elektronický podpis časového razítka.
- Položka Rozšíření je určena pro dodatečné informace, jež mohou být v budoucnu do časového razítka přidávána.

8.5 Ověřování časového razítka

Časové razítko je datová struktura digitálně podepsaná TSA, která časové razítko vydala. Takže ověřování tohoto podpisu provedeme pomocí certifikátu TSA. Jenže certifikát TSA vydala nějaká důvěryhodná certifikační autorita (Obrázek 8.5). Vzniká tak řetězec až k důvěryhodné kotvě.

Představme si, že jsme již ověřili platnost certifikátů TSA v řetězci certifikátů až k důvěryhodné kotvě. A to s výsledkem, že certifikát TSA je platný. Pak pomocí takto ověřeného certifikátu můžeme ověřit elektronický podpis časového razítka. Je-li platný elektronický podpis časového razítka, musíme ověřit otisk dokumentu (*message imprint*). Tj. musíme spočíst otisk z originální zprávy a ověřit, zdali se shoduje s otisk uvedeným v časovém razítku.

Nyní, když máme kryptografickou gymnastiku za sebou, zjistíme, jestli je pro nás TSA důvěryhodná. Ta by nás o své důvěryhodnosti měla přesvědčit pomocí politiky uvedené v časovém razítku …

8.6 Platnost časového razítka

Z hlediska certifikační autority je TSA speciálním případem koncového uživatele. Certifikační autorita vydává certifikát TSA s několika odlišnostmi od certifikátů pro běžné koncové uživatele.

Certifikát TSA je odlišný od certifikátů ostatních koncových uživatelů nejenom tím, že má naplněna rozšíření tak, aby nešel použít k ničemu jinému než verifikaci časových razítek, ale zejména tím, že mívá odlišnou periodu, po kterou je platný. Certifikát TSA zpravidla mívá periodu platnosti obdobnou jako certifikát CA, která jej vydala.

Časové razítko je digitálně podepsaná struktura. Platnost této struktury je tak omezena platností certifikátu, pomocí kterého její elektronický podpis ověřujeme, tj. platností certifikátu TSA. Pokud by tomu tak nebylo, pak bychom měli potíže s ověřováním časových razítek z elektronických podpisů.

Představme si, že ověřujeme elektronický podpis opatřený časovým razítkem z elektronického podpisu (Obrázek 8.6). Pak musíme:

- Verifikovat časové razítko z elektronického podpisu pomocí příslušného certifikátu TSA.
- Verifikovat elektronický podpis pomocí příslušného certifikátu koncového uživatele, který podpis vytvořil.

Pokud by se doba platnosti certifikátu koncového uživatele překrývala s dobou platnosti certifikátu TSA např. jen 1 den, pak by se musel obnovit elektronický podpis dokumentu již po jednom dni. Snahou tedy je, aby doba platnosti certifikátu TSA byla větší než platnost certifikátů běžných koncových uživatelů. S podobným problémem jsme se již setkali při obnovování certifikátů certifikačních autorit, které se musí obnovovat s dostatečným předstihem, aby nenastala situace, že certifikát koncového uživatele bude platný ještě v době, kdy už vypršela platnost certifikátu CA, která jej vydala.

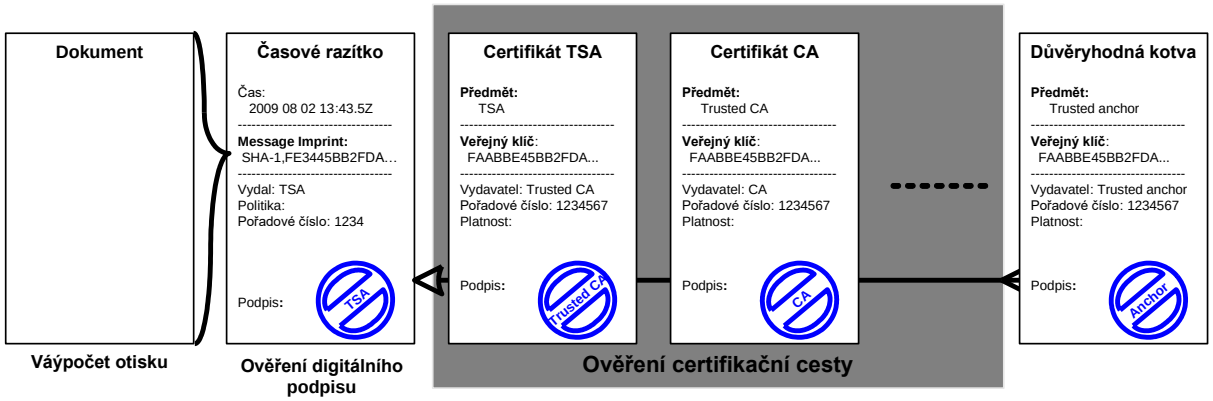
Obdobně i v případě TSA vydáme nový certifikát TSA s takovým předstihem aby:

- v době, kdy se platnost starého a nového certifikátu TSA překrývá, bylo technicky možné veškerá časová razítka vytvořená starou TSA obnovit pomocí nové TSA.
- všechna časová razítka elektronických podpisů dokumentů vytvořená pomocí starého certifikátu TSA byla platná alespoň po dobu platnosti všech uživatelských certifikátu určených pro verifikaci elektronických podpisů dokumentů.

8.7 Co časové razítko není

Časové razítko (ve tvaru podle RFC-3161) není samospasitelné. Je důkazem, že otisk z nějakého dokumentu existoval v daném čase. Z toho lze odvozovat, že i dokument musel existovat před časem uvedeným v časovém razítku. Časové razítko však neobsahuje identifikaci držitele dokumentu, takže časové razítko nemůže sloužit jako důkaz, že nějaký dokument byl vlastněn konkrétní osobou v daném čase.

Základním problémem časového razítka je též jeho doba platnosti. Na první pohled to totiž vypadá, že časové razítko je trvale platné. Avšak není tomu tak. Časové razítko je digitálně podepsaná struktura, kterou podepisuje TSA. Tento elektronický podpis se verifikuje pomocí certifikátu TSA a ten má omezenou dobu platnosti. Doba platnosti časového razítka je tak nepřímo dána dobou platnosti certifikátu TSA.



Obrázek 8.6 Verifikace časového razítka

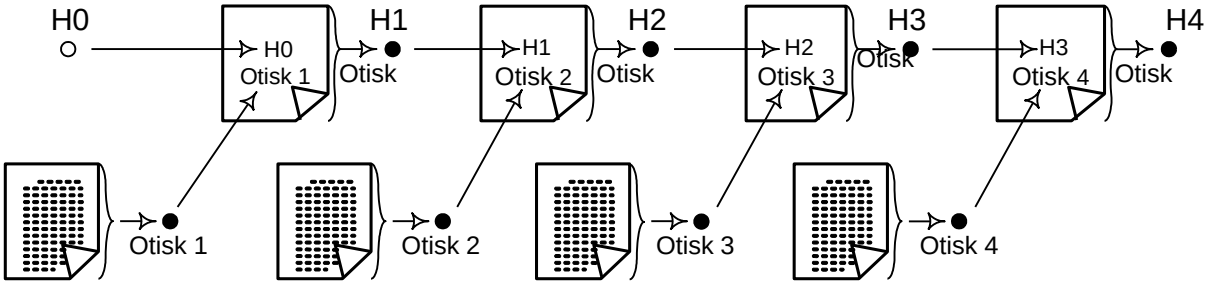
9 ERS

9.1 Provázané otisky

Časové razítko obsahovalo absolutní čas UTC, podle kterého bylo možné odvodit, kdy dokument existoval. I když čas vyjadřuje posloupnost jednotlivých dějů, tak ani přesný UTC v časovém razítku není zcela dokonalý k vyjádření této posloupnosti.

Mechanismus provázaných otisků se neorientuje na svázání času s dokumentem. Jeho cíl je prostší, zabývá se relativní časovou autentizací (*Relative Temporal Authentication) - RTA*. Při relativní časové autentizaci, na rozdíl od asociace události s právě aktuálním časem, je událost asociována s událostmi, které nastaly těsně před a po naší události. Neřekneme, že dokument byl orazítkován v 21:18:26.34, nýbrž řekneme, že dokument byl orazítkován po orazítkování předchozího dokumentu a před orazítkování následujícího dokumentu. Pokud požadavky na razítkování dokumentů docházejí hojně a přitom náhodně, pak je tato časová posloupnost dokumentů dosvědčitelná a není žádná možnost, jak tyto události ovlivnit.

9.1.1 Lineární schéma



Obrázek 9.1 Lineární linkovací schéma tvoří orientovaný graf

Princip provázaných otisků se nejlépe pochopí na lineárním linkovacím schématu (Obrázek 9.1).

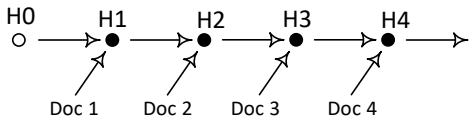
Cílem je zajistit důkaz, že Dokument 1 existoval před Dokumentem 2 a ten před Dokumentem 3. Tj. vytvořit důkaz o tom, že Dokumenty 1, 2, 3, 4, … tvořily časovou posloupnost. Cílem není dokázat, že Dokument 2 existoval v 21:18:26.34, ale jen že existoval po tom, co se razítkoval Dokument 1 a před tím než se razítkoval Dokument3.

Mechanismus je velice jednoduchý (sledujte Obrázek 9.1):

1. Na prvopočátku pevně zvolíme algoritmus ***h*** pro výpočet otisku, který budeme používat ve všech dalších krocích.
2. Zvoleným algoritmem spočteme otisk H0 z libovolného dokumentu. *(Tento krok je potřebný pouze formálně, vytváříme si tak startovací bod.)*
3. Z Dokumentu 1 spočteme otisk 1. Nyní vytvoříme umělý dokument, který vznikne spojením H0 a otisku 1. Z toho umělého dokumentu spočteme otisk H1. H1 tak vznikl z H0 a Dokumentu 1. Teď to nejdůležitější: abychom mohli H1 vytvořit, musel před jeho vytvořením existovat jak H0 tak i Dokument 1.
4. Obdobně postupujeme i s dalšími dokumenty: z Dokumentu n spočteme otisk n, který spolu s H(n-1) vložíme do umělého dokumentu a spočteme z něj otisk Hn. Všechny Dokumenty 1,2,3,…n-1 tedy musely existovat před tím, než byl vytvořen Hn.

Jak nyní postupovat při ověření dokumentu Doc 2? Zdánlivě jednoduše. Stačí najít držitele dokumentů Doc1 a Doc3 a přivést je jako svědky. Doc2 musel přece existovat mezi tím, kdy držitel Doc1 a držitel Doc3 žádali o orazítkování svých dokumentů. Potíž je v tom, že zajištění držitelů dokumentů Doc1 a Doc3 nebude jednoduché.

Otisk může být publikován (Obrázek 9.5) v novinách, ale lze jej doplňkově publikovat i v elektronický podobě jednoduše tak, že se vydá ve tvaru klasického časového razítka obsahujícího např. týdenní otisk.



Obrázek 9.2 je poměrně komplikovaný, proto si jej zjednodušíme na graf

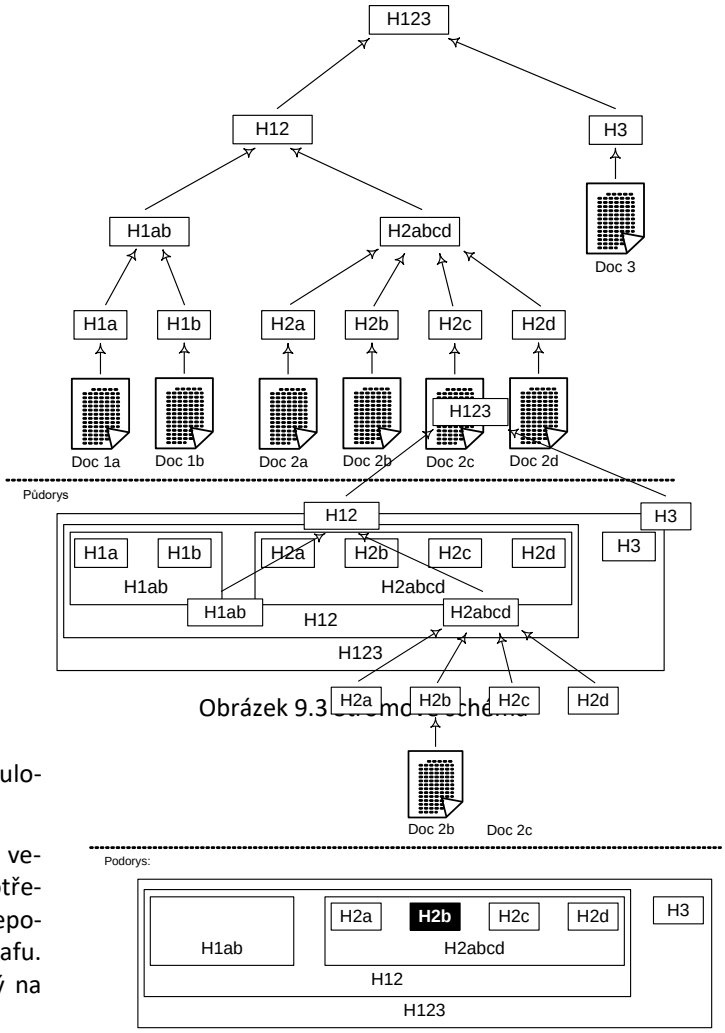
Největším nezbezpečním při lineárním linkování dokumentů je ztráta některého z otisků v řadě.

Lineární linkovací schéma vypadá logicky, když si nakreslíme graf o šesti uzlech. V případě, že se linkuje velké množství dokumentů, potřebujeme pro jejich ověřování počítat velké množství otisků. Toto množství lineárně roste se vzdáleností od publikovaného otisku. V praxi je tedy těžko využitelné. U následujícího stromového linkování je tento nárůst pouze logaritmický.

9.1.2 Stromové schéma

Představme si, že dokumenty nevznikají po jednom, ale po skupinách. Na obrázku Obrázek 9.3 máme tři skupiny dokumentů: (Doc1a , Doc1b), (Doc2a, Doc2b, Doc2c, Doc2d) a samostatný dokument Doc3. Z jednotlivých dokumentů spočteme otisky: H1a, H1b, H2a, H2b, H2c, H2d a H3.

Nyní otisky jednotlivých dokumentů spojíme do umělých dokumentů, ze kterých spočteme kumulované otisky: H1ab (dokument tvořený spojením H1a a H1b), H2abcd, H3. Nyní zkumulujeme H1ab s H2abcd do výsledného otisk H12. Kumulací H12 a H3 vznikne kořenový otisk H123, který může být veřejně publikován. Stromové schéma vidíme na obrázku 9.3.



Obrázek 9.3 Stromové schéma

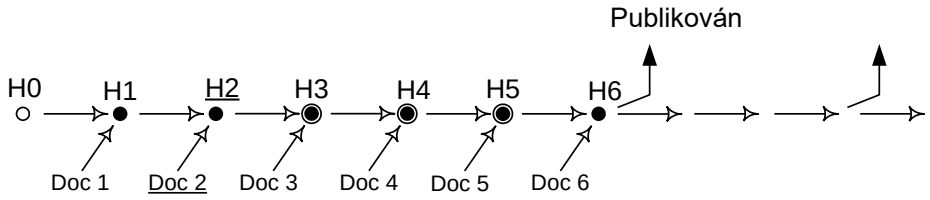
Někdy se stromová struktura nekreslí jako stromový graf, ale jako jakýsi půdorys tohoto grafu vyjadřující jak jsou jednotlivé otisky kumulovány.

Předpokládejme, že kumulovaný otisk H123 byl veřejně publikován. Nyní budeme zkoumat, co potřebujeme k ověření dokumentu Doc2b. Zdaleka nepotřebujeme všechny listy našeho stromového grafu. Stačí nám totiž jen redukovaný graf zobrazený na obrázku Obrázek 9.4.

Obrázek 9.4 Redukovaný strom pro verifikaci Doc2b

Pro ověření rozšířeného časového razítka tak potřebujeme některé otisky a informaci jak je kumulovat. Tato informace může být vyjádřena pseudo jazykem:

SEQ (SEQ (**H1ab**, SEQ (**H2a, H2b, H2c, H2d**)), **H3**)



Obrázek 9.5 čas od času se otisk publikuje

Ještě zajímavější je co potřebujeme pro verifikací Doc3. Jak je znázorněno na obrázku Obrázek 9.4, tak nám stačí SEQ (**H12**,**H3**).

Výsledkem je tak redukovaný (nebo též Merklův) strom provázaných otisků, který z pohledu konkrétního listu stromu obsahuje jen ty otisky, které jsou nutné pro verifikaci konkrétního otisku ke kořeni stromu.

9.1.3 ERS

Cílem standardu ERS (*Evidence Record Syntax*), který vyšel jako RFC 4998, je vytvořit dlouhodobý důkaz, že dokument existovat v jistém čase a od té doby nebyl změněn. Tímto důkazem je ERS záznam (*Evidence Record Syntax*). ERS záznam můžeme vytvořit k samotnému dokumentu nebo ke skupině dokumentů.

ERS záznam může udržovat paralelně s archivovaným dokumentem nebo v případě digitálně podepsaných dokumentů může být též přidán jako nepodepisovaný atribut elektronického podpisu.

ERS záznam používá jako důkaz existence dokumentů v čase redukovaný strom provázaných otisků (Merklův strom). Tuto myšlenku rozpracovává a formalizuje tak, že definuje tzv. Archivní časové razítko[*)], které obsahuje strom provázaných otisků a klasické časové razítko z kořenového otisku téhož stromu. Jedná se o obdobu Obrázek 9.4

Někdy se zajištění ERS obchodně označuje jako softwarový ROM.

[*)] Nezaměňovat s archivním časovým razítkem používaným v případě podpisů AdES

10 CMS

CMS (*Cryptographic Message Syntax*) je standard specifikující zabezpečení datové zprávy, který byl původně zaveden RSA Security Inc. jako standard PKCS#7.

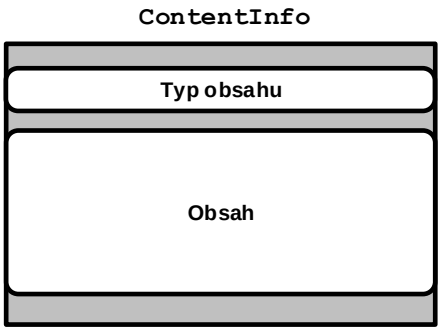
Datovou zprávou rozumíme blok nebo soubor dat, který má jasný začátek a konec. CMS tedy neslouží k zabezpečení toku dat (tok dat zabezpečuje např. protokol TLS). Na rozdíl od podpisů DSIG a XAdES standard CMS nevidí do vnitřní struktury zabezpečované datové zprávy – bere datovou zprávou jako černou skříňku, jako řetězec bajtů. Zabezpečení zprávy chápe CMS kompletně – řeší nejenom elektronický podpis, ale např. i šifrování zprávy. Hovoříme pak o jednotlivých typech CMS zpráv pro jednotlivá zabezpečení.

Tabulka 10.1 Typy CMS zpráv

Typ zprávy	Význam
Data	Data (zpravidla typ zprávy vstupující do zabezpečení)
SignedData	Elektronicky podepsaná data.
EnvelopedData	Data v elektronické obálce – data šifrovaná tajným klíčem, který je přidán ke zprávě zašifrován veřejnými klíči adresátů. Používá se např. pro šifrování elektronické pošty.
DigestedData	Data a jejich otisk (s využitím se autor článku nesetkal).
EncryptedData	Data šifrovaná jiným způsobem – např. pro zabezpečení kryptografického materiálu na disku.
AuthenticatedData	Autentizavaná data (s využitím se autor článku nesetkal).

Výsledkem zabezpečení datové zprávy protokolem CMS je vždy struktura ContentInfo (CMS zpráva), která je sekvencí dvou položek (Obrázek 9.1), a to typu obsahu a vlastního obsahu, kterým může být elektronicky podepsaná zpráva, zpráva v elektronické obálce apod.

Díky této konstrukci lze přečtením první položky každé CMS zprávy vždy jednoduše zjistit, o jaký typ obsahu se jedná (zkráceně: o jakou CMS zprávu se jedná), tj. zda se jedná o elektronicky podepsanou



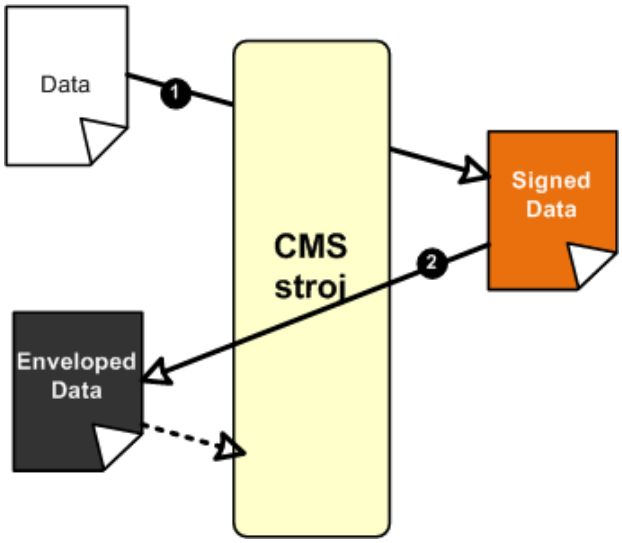
Obrázek 9.1 Formát CMS zprávy

zprávu, šifrovanou zprávu apod. Jednotlivé typy CMS zpráv jsou uvedeny v tabulce Tabulka 10.1.

Jak CMS pracuje? Asi nejlépe demonstrující představou je představa CMS stroje. Na obr. Obrázek 10.2 je znázorněn CMS stroj, do kterého vstupuje nezabezpečená zpráva (obsah) a vystupuje z něj zabezpečená zpráva, přitom způsob zabezpečení je na uživateli. CMS stroj uživateli nabízí: elektroní podpis, elektroní obálku, otisk ze zprávy, autentizaci zprávy či prosté zašifrování zprávy.

Mnohdy však nestačí zprávu jen podepsat či šifrovat, ale chceme jim jak podepsat, tak i šifrovat. Pak CMS stroj voláme několikrát za sebou (Obrázek 10.2). V případě bezpečných konferencí se CMS stroj volá dokonce třikrát.

Architektura CMS je natolik obecná, proto se mnohé současné protokoly ani zabezpečením obsahu svých



Obrázek 10.2 CMS stroj

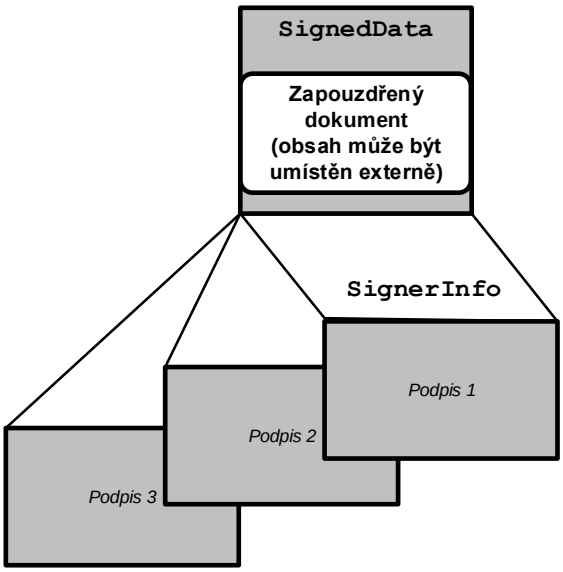
zpráv nezabývají – uvedou jen, že zabezpečení ponechávají na CMS stroji. Tuto filosofii zastává např. RFC-3161 specifikující časová razítka.

10.1 Elektronicky podepsaná data (Signed-Data)

Pro elektronický podpis standard CMS využívá typ SignedData (Obrázek 10.3). Struktura SignedData v sobě zahrnuje množinu podpisů – množinu struktur SignerInfo. Každý prvek této množiny obsahuje jeden (paralelní) podpis.

Detailní popis struktury SignedData je pak schématický znázorněn na Obrázek 10.3. Význam jednotlivých položek je následující:

- Položka Verze obsahuje verzi struktury elektronicky podepsané zprávy. Původní standard PKCS#7 používá verzi 1, dnes existuje celá řada verzí.
- Položka „Množina použitých algoritmů Otisk“ obsahuje množinu identifikátorů algoritmů použitých pro výpočet otisku.
- Položka „Zapouzdřený dokument“ může obsahovat podepisovaná data, tj. např. CMS zprávu typu Data. Důležité je slovo „může“. V případě elektronického podpisu totiž rozlišujeme externí a interní elektronický podpis:
    - V případě interního elektronického podpisu je obsah zprávy umístěn v této položce.
    - V případě externího podpisu je obsah zabezpečené zprávy umístěn externě, např. v jiném souboru.
      Prakticky to znamená, že pokud je zpráva uložena v externím souboru, máme podpis zprávy uložen v jiném samostatném souboru. V případě elektronicky podepsaného e-mailu (S/MIME) se e-mail skládá ze dvou dílčích částí: vlastní zprávy a elektronického podpisu. Výhodou externích podpisů je, že původní zprávu je možné snadno zobrazit (zpracovat) i pomocí software, který elektronický podpis nepodporuje.



Obrázek 10.3 Struktura SignedData

- Položka Certifikáty obsahuje množinu certifikátů, které budou užitečné při verifikaci elektronických podpisů, tj. k tomu, aby sestavil certifikační cestu až k důvěryhodné kotvě. Nikde však není řečeno, že by tam musely být všechny certifikáty, které uživatel bude potřebovat pro verifikaci elektronického podpisu, a mohou tam být i certifikáty, které uživatel vůbec nepotřebuje. Při tvorbě takovéto zprávy je dobré si uvědomit, že ji bude nutné ověřovat třeba za 10 let a mnohé certifikáty v té době již prošlé může být obtížné obstarávat.
- Položka „Revokační informace“ může obsahovat informace platnosti certifikátů (např. CRL). O CRL platí totéž, co bylo uvedeno u certifikátů. Opět je třeba vžít se do role toho, kdo bude elektronický podpis po letech ověřovat, a tudíž bude hledat příslušná CRL platná v době podepsání zprávy. I kdyby certifikáty byly ještě platné, tak v nich uvedené rozšíření odkazuje na aktuální CRL, nikoliv na CRL platné v okamžiku podpisu!

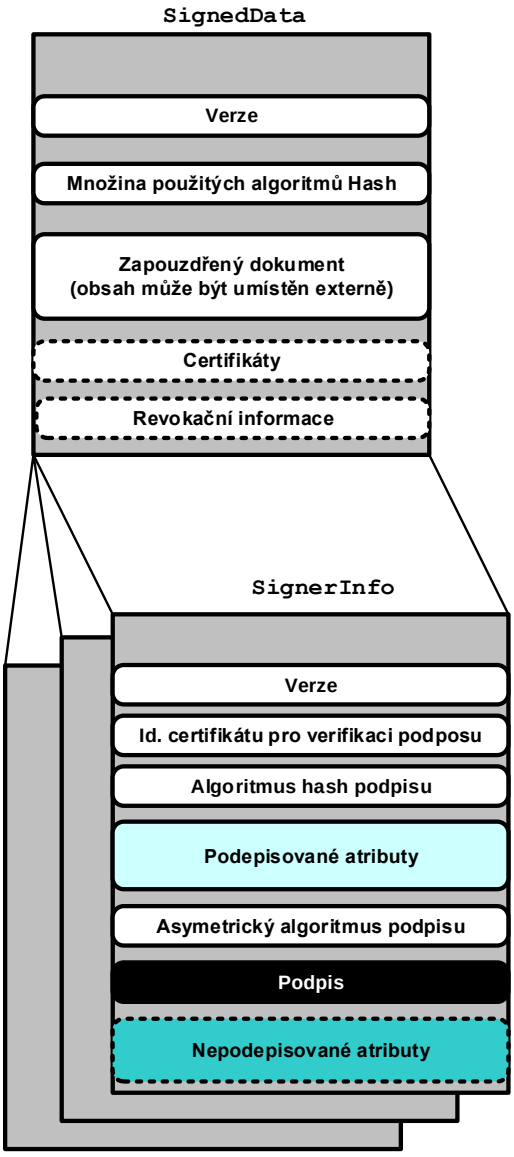
- Množinu elektronických podpisů. Už, už jsem chtěl napsat „… obsahuje množinu elektronických podpisů vstupních (zapouzdřených) dat“, což by nebylo správně, protože to by mnohdy bylo málo. Elektronický podpis nemusí totiž být počítán nejen ze vstupních dat, ale též i např. z aktuálního data a času – obecněji z tzv. podepisovaných atributů. Každý podpis je typu SignerInfo. Množina může být i prázdná v případě tzv. degenerované zprávy SignedData, které se používají k distribuci množiny certifikátů ve formátu .p7b.

Obecně může zpráva SignedData obsahovat více elektronních podpisů. To je běžné i u rukou psaného podpisu. Např. pro platební příkazy nad stanovený limit mohou být bankou požadovány také dva podpisy. Ještě běžnější je podepisování smluv. Smlouva je vztah minimálně mezi dvěma stranami, takže na smlouvě budou také minimálně dva podpisy atd.

10.1.1 Vlastní elektronický podpis

Vlastní elektronický podpis (SignerInfo) se skládá z následujících položek:

- Položka Verze obsahuje verzi této datové struktury.
- Položka „Id. certifikátu pro verifikaci podpisu“ obsahuje identifikaci certifikátu pro verifikaci tohoto elektronického podpisu. Identifikací může být buď dvojice předmět certifikátu + pořadové číslo certifikátu nebo identifikátor klíče předmětu.
- Položka “Algoritmus otisk podpisu“ obsahuje algoritmu otisku, kterým je spočten otisk dokumentu pro elektronický podpis.
- Položka „Podepisované atributy“ obsahuje další informace, které budou zahrnuty do výpočtu elektronického podpisu.
- Položka „Asymetrický algoritmus podpisu“ definuje asymetrický algoritmus použitý k výpočtu elektronického podpisu včetně jeho příslušných parametrů. Windows nazývají tuto položku „Algoritmus podpisu“.
- Položka Podpis obsahuje příslušný elektronický podpis, tj. asymetrickou šifrou šifrovaný otisk. Asi největším překvapením je způsob, jakým se počítá otisk pro vytvoření elektronického podpisu. Problém je totiž v tom, že je nutné spočíst otisk nejenom ze samotných vstupních dat, ale i z podepisovaných atributů. Proto se rozlišují dva případy a v každém se otisk počítá z něčeho jiného:



Obrázek 10.4 Struktura SignedData podrobněji

  - CMS zpráva neobsahuje žádný podepisovaný atribut. V tomto případě se otisk spočte ze vstupních dat.
  - Zpráva obsahuje podepisované atributy. V tomto případě se otisk nepočítá přímo ze vstupních dat, ale z podepisovaných atributů. Nejprve se totiž spočte otisk z podepisované zprávy a ten se vloží do podepisovaného atributu „Otisk zprávy (někdy též označovaný jako Výtah ze zprávy)“. Následně se pak ze všech podepisovaných atributů spočte otisk, ze kterého se teprve až vytváří elektronický podpis. Otisk podepisované zprávy je tak nepřímo obsažen v otisku, ze kterého se vytváří elektronický podpis. V tomto případě musí být jako po-

depisované atributy uvedeny alespoň atributy: „Otisk zprávy (někdy též označovaný Výtah ze zprávy)“ a „Typ obsahu“.

- Volitelná položka „Nepodepisované atributy“ je určena pro nepodepisované atributy elektronického podpisu. Tj. atributy, které se nezahrnují do výpočtu otisku, ze kterého se počítá elektronní podpis. Syntaxe je obdobná podepisovaným atributům. Klasickým nepodepisovaným atributem je „Kontrasignatura“.

Zastavme se u podepisovaných a nepodepisovaných atributů. Podpis CMS zprávy se zpravidla nepočítá z hodnoty otisku samotného podepisovaného dokumentu, ale z otisk podepisovaných atributů. Avšak jedním z podepisovaných atributů je otisk dokumentu.

A co nepodepisované atributy? Klasickým případem nepodepisovaného atributu je kontrasignatura. Kontrasignaturu známe z praxe. Když prezident s cizím prezidentem podepíší nějakou mezinárodní smlouvu (připojí paralelní podpisy), pak smlouva zpravidla ještě není platná. Platnou se stane, až ji schválí parlamenty, tj. až předsedové parlamentu kontrasignují podpisy svých prezidentů, tj. až ke každému paralelnímu podpisu sériově připojí svůj podpis.

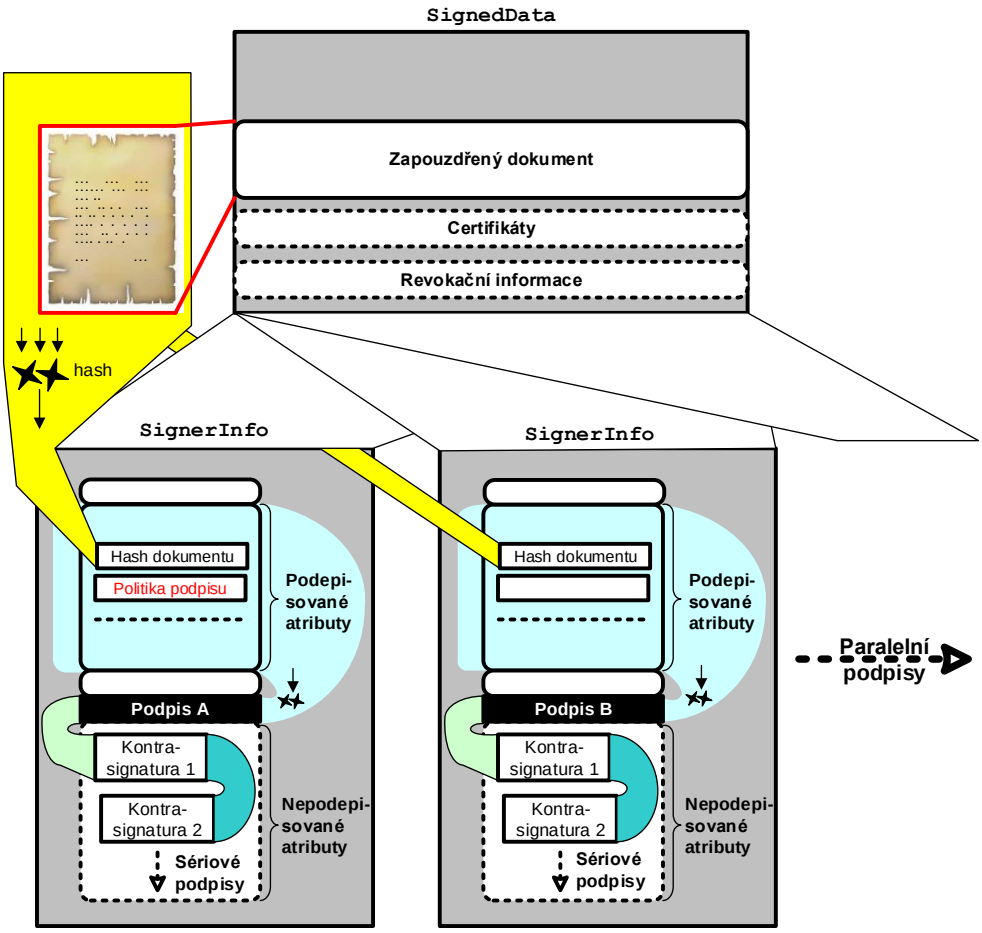
V případě CMS. Každý (paralelní) podpis může obsahovat nepodepisované atributy Kontrasignatura, tj. může obsahovat další sériové podpisy (podpisy podpisů).

Atribut	**Význam**	**Podepisovaný/nepodepisovaný**
Typ obsahu (*Content Type*)	Specifikuje typ obsahu podepisované nebo autentizované zprávy.	Podepisovaný nebo autentizovaný
Kontrasignatura (*Countersignature*)	Obsahuje další sériový elektroní podpis zprávy	Pouze nepodepisovaný
Otisk zprávy - též „Výtah ze zprávy“ (*Message Digest*)	Obsahuje otisk podepisované zprávy	Podepisovaný nebo autentizovaný
Čas podpisu (*Signing Time*)	Specifikuje čas, kdy dokument byl podepsán. Nejedná se ale o časové razítko! Tento čas podepisující osoba může libovolně nastavit, neboť se bere ze systémového času počítače!	Podepisovaný nebo autentizovaný
Schopnosti protokolu S/MIME (*S/MIME Capabilities*)	Specifikuje algoritmy podpisu, symetrické šifrovací algoritmy a další schopnosti preferované software, kterým byla CMS zpráva vytvořena.	Pouze podepisovaný
Šifrovací certifikát (*sMIMEEncryptionKeyPreference*)	Specifikuje certifikát, jež má být použit pro šifrování odpovědi.	Pouze podepisovaný
Content Hints	Poskytuje informace o zapouzdřené zprávě (vhodné zejména pro případ vícenásobného zapouzdření, které zamezuje k přístupu k Předmětu hluboce zapouzdřené zprávy).	obojí
Content Identifier	Obsahuje jednoznačný identifikátor zprávy (vhodné např. pro párování doručeny s doručenou zprávou)	obojí
Content Reference	Obsahuje odkaz jedné CMS zprávy na jinou (např. odkaz v odpovědi na původní zprávu).	Pouze podepisovaný
Equivalent Label	Bezpečnostní návěští	Pouze podepisovaný
ESS SecurityLabel		Pouze podepisovaný

Atribut	Význam	Podepisovaný/nepodepisovaný
Message Signing Digest	Obsahuje otisk z podepisovaných atributů původní zprávy (využívá jej doručenka).	Pouze podepisovaný
ML Expansion History	Obsahuje seznam bezpečných konferencí, kterým byl zpráva již distribuována (opatření proti zacyklení konferencí)	Pouze podepisovaný
Recept Request	Tímto atributem žádáme adresáta o doručenku. Doručenku následně vrací adresát po úspěšném ověření elektroního podpisu zprávy.	Pouze podepisovaný
Signing Certificate	Umožňuje do podpisu zahrnout i identifikaci certifikátu pro verifikaci podpisu.	Pouze podepisovaný
Other signing certificate	Stejný význam jako atribut *Signing Certificate*, ale umožňuje použít též jiný otisk než SHA-1	Pouze podepisovaný
Signature policy identifier	Obsahuje politiku elektroního podpisu.	Pouze podepisovaný
Commitment Type Indication	Specifikuje odpovědnost osoby za svůj podpis pod dokumentem	Pouze podepisovaný
Signer location	Slouží pro specifikaci adresy podepisovaného.	Pouze podepisovaný
Signer attributes	Slouží pro specifikaci role (pracovní pozice) podepisovaného	Pouze podepisovaný
Content timestamp	Obsahuje časové razítko z dokumentu (nikoliv z podpisu).	Pouze podepisovaný

Atribut	Význam	Podepisovaný/nepodepisovaný
Signature timestamp	Obsahuje časové razítko z podpisu.	Nepodepisovaný
Complete Certificate Refs	Obsahuje identifikace všech certifikátů certifikačních autorit, které mají být použity pro verifikaci elektroního podpisu.	Nepodepisovaný
Complete revocation Refs	Obsahuje úplnou množinu odkazů na revokační informace nutné jak pro ověření certifikátu uživatele, který podpis vytvořil, tak i certifikátů všech CA nutných k jeho ověření.	Nepodepisovaný
Attribute certificate references	Obsahuje úplnou množinu odkazů na certifikáty atributových autorit, které jsou třeba pro ověření atributových certifikátů potřebných k verifikaci tohoto podpisu.	Nepodepisovaný
Attribute revocation references	Obsahuje úplnou množinu odkazů na revokační informace (ACRL, OCSP atd.) nutných pro verifikaci atributových certifikátů.	Nepodepisovaný
Certificate values	Obsahuje úplnou množinu certifikátů potřebných pro verifikaci podpisu	Nepodepisovaný
Revocation values	Obsahuje úplnou množinu revokačních informací nutných pro verifikaci certifikátu podepisovaného i certifikátů všech příslušných CA.	Nepodepisovaný

Jiná situace je, když se vytváří kontrasignatura. Kontrasignatura je podpis z podpisu. Kontrasignatura se



Obrázek 10.5 Násobné podpisy

Atribut	Význam	Podepisovaný/nepodepisovaný
ES-C Timestamp	Obsahuje ES-C časové razítko.	Nepodepisovaný
Time-Stamped certificates and CRL		Nepodepisovaný
Archive Timestamp	Obsahuje archivní časové razítko	Nepodepisovaný

10.1.2 Paralelní a sériový podpis

Mnohé dokumenty (např. smlouvy) podepisuje více stran. Tj. každá strana připojí paralelně pod dokument svůj podpis (svou strukturu `SignerInfo`). Paralelní podpisy jsou na sobě nezávislé a v případě CMS hrozí i zmíněné nebezpečí odříznutí jednoho z podpisů zobrazené na obr. Obrázek 10.7.

proto též označuje jako sériový podpis. Kontrasignatura je nepodepisovaným atributem, proto také může být útočníkem odstraněna, aniž by se to dalo na první pohled poznat. U sériově podepsané zprávy (kontrasignatury) útočník nemůže přehodit její pořadí kontrasignatury s předchozí kontrasignaturou aniž by se na to přišlo.

10.1.3 Ověřování podpisu

Slovo ověřování elektronického podpisu je dvojznačné:

- Může označovat notářský výkon. Tj. činnost třetí stany (Důvěryhodného poskytovatele ověření podpisu), který stvrzuje pravost podpisu.
- Může označovat formální ověřování elektronického podpisu od důvěryhodné kotvy. Tímto typem ověřování se dále budeme zabývat v této kapitole.

Zdánlivě je ověření elektronického podpisu snadné. Jelikož pod zprávou máme několik na sobě nezávislých podpisů, můžeme je ověřovat na sobě nezá-

visle. Ověřování podpisu se tak převádí na ověřování každého jednotlivého podpisu samostatně - jakoby zprávy s jedním podpisem (Obrázek 10.6).

Obrázek 10.6 Verifikace elektronického podpisu

Ověření podpisu pak předpokládá:

1. Ověření certifikátu určeného pro verifikaci tohoto podpisu až k důvěryhodné kotvě (ověření certifikátu podepisované osoby).
2. Verifikace samotného podpisu:
   a. Spočtení otisku zprávy, porovnání spočteného otisku s otiskem uvedeným v podepisovaném atributu Otisk zprávy. Důvodem je skutečnost, že příjemce nesmí automaticky věřit otisku zprávy zaslaného odesílatelem.
   b. Dešifrování elektronického podpisu veřejným klíčem podepisovaného z jeho v předchozím bodě verifikovaného certifikátu.
   c. c.A konečně porovnání veřejným klíčem dešifrovaného výsledku se spočteným otiskem ze zprávy.

Jenže takováto mechanická verifikace elektronického podpisu je vhodná nanejvýš pro verifikace běžné elektronický podepsané e-mailové korespondence.

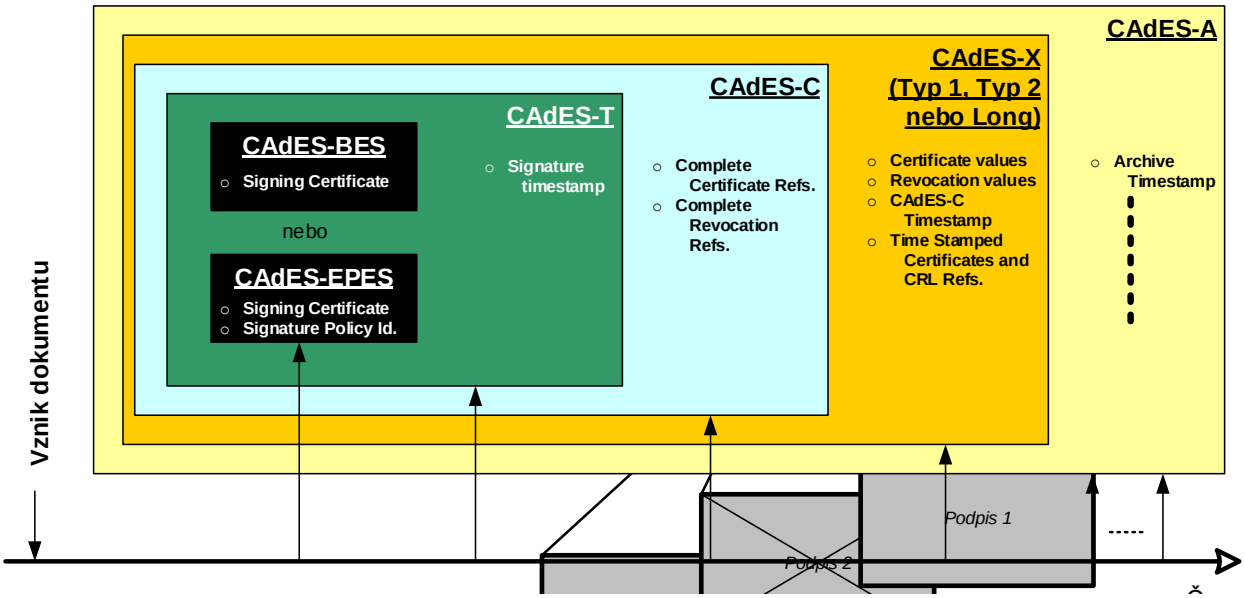
Je třeba si uvědomit, že dokument může mít více podpisů včetně kontrasignatur (Obrázek 10.5). Je dokument platný, když selže ověření jen některého z podpisů? A i když formální verifikace všech elektronických podpisů dopadne úspěšně. Je pak dokument platný? Byly tyto osoby vůbec oprávněny dokument podepsat? Tyto otázky samotný standard CMS přímo neřeší.

10.1.4 Export certifikátu

Degenerovaným případem elektronicky podepsané CMS zprávy je zpráva neobsahující žádný elektronický podpis (žádnou strukturu SignerInfo), tj. CMS zpráva obsahující pouze certifikáty a případně ještě CRL. Tento typ podepsané zprávy se používá k distribuci certifikátu. Na rozdíl od ostatních formátů souborů nesoucích certifikáty má tato degenerovaná CMS zpráva jednu podstatnou výhodu. Nemusí totiž obsahovat pouze jeden certifikát, ale může obsahovat celou množinu certifikátů (včetně atributových certifikátů).

Ve Windows můžeme certifikát exportovat několika způsoby:

- Do degenerované elektronicky podepsané zprávy CMS (jak je na obrázku přímo napsáno „PKCS č. 7“ čímž je asi míněn standard PKCS#7 – předchůdce standardu CMS). Tato možnost umožňuje exportovat množinu certifikátů.
- Volbou „Binární X.509, kódování DER“ získáme binární DER kódovaný soubor.
- Volbou „X.509, kódování Base64 (CER)“ získáme formát PEM (Base64 kódován DER).
- Další možností je exportovat certifikát i se soukromým klíčem. Exportujeme tak dvojici certifikát a příslušný soukromý klíč. Formát takového výstupního souboru je specifikován normou PKCS#12 (tento formát se označuje “Personal Information Exchange” – PFX). Výstupní soubor formátu PKCS#12 má zpravidla příponu .pfx nebo p12. Pro vytvoření PFX souboru je nutné, aby soukromý klíč byl exportovatelný. V souboru je privátní klíč uložen většinou v šifrované podobě.



Obrázek 10.8 Hierarchie podpisů CAdES s klíčovými atributy

10.1.5 Nevýhody podpisu CMS

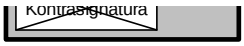
CMS je sice patrně nejrozšířenějším typem elektronického podpisu, přesto trpí jistými neduhy:

1. Některé podpisy je možné ze zprávy odstranit, aniž by to bylo na první pohled zjistitelné (Obrázek 10.7). Dokument se přitom technicky nadále může jevit platným.
2. Některé části struktury SignedData (certifikáty, revokační informace, identifikátor certifikátu pro verifikaci podpisu atd.) nejsou zahrnuty do otisku podpisu, tj. nejsou podpisem zabezpečeny. Útočník je může snadno vyměnit (resp. odstranit).
3. Čas podpisu je sice podepisovaným atributem, ale zpravidla se bere jako systémový čas počítače, takže je jen informativní.
4. Certifikát pro ověření podpisu vyprší nebo je odvolán. Budeme poté dokument brát jako platný?

10.1.6 CAdES

Standard CAdES (*CMS Advanced Electronic Signatures*) se pokouší zmíněné neduhy řešit. S výjimkou bodu 1 se mu to úspěšně daří. Řeší to přidáním dalších podepisovaných a zejména nepodepisovaných atributů.

Standard CAdES vychází ze standardu CMS. Specifikuje, které podepisované a které nepodepisované atributy má zaručený podpis obsahovat. Standard specifikuje několik formátů podpisu, které tvoří hierarchii od nejjednoduššího CAdES-BES až po komplexní CAdES-A (zcela analogicky, jak jsme se s tím setkali v minulém dílu seriálu u podpisu XAdES).



Obrázek 10.7 Útok odříznutím podpisu

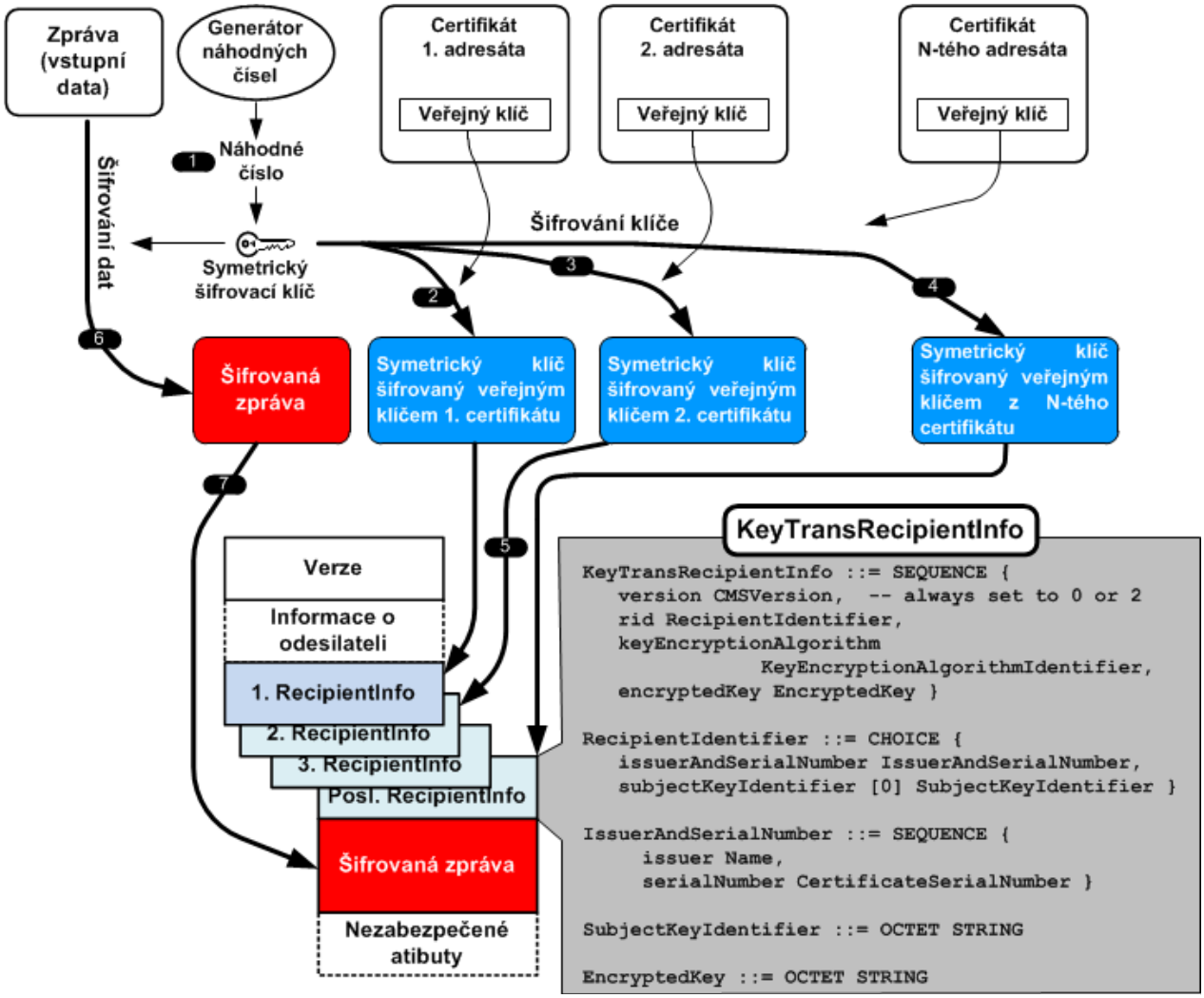
Tato hierarchie se nejčastěji zakresluje pomocí vnoření (Obrázek 10.8).

Jádrem hierarchie podpisů jsou prvotní formáty podpisů:

- Nejjednodušším prvotním formátem je CAdES-BES, který je vlastně formátem CMS jen s tím, že vyžaduje, aby identifikace podpisového certifikátu byla zajištěna podpisem, tj. vyžaduje některý z podepisovaných atributů, které obsahuje identifikaci podpisového certifikátu.
- Dalším prvotním formátem je CAdES-EPES, který navíc oproti CAdES-BES vyžaduje podpisovaný atribut obsahující identifikátor politiky elektronického podpisu. O politice elektronického podpisu byla zmínka již v minulém dílu tohoto seriálu.

V rámci hierarchie jsou specifikovány i další formáty, které ale vznikají až při ověřování podpisu nafukováním nepodepisovaných atributů:

- CAdES-T, který navíc obsahuje časové razítko z podpisu, tj. konzervuje podpis v čase. Vypršení platnosti nebo odvolání podpisového certifikátu po vložení tohoto časového razítka tak již nemůže mít vliv na platnost podpisu.
- CAdES-C, který navíc obsahuje identifikace všech informací nutných pro ověření podpisu.



Obrázek 10.9 Elektronická obálka (EnvelopedData)

- CAdES-X Long, který obsahuje veškeré informace nutné pro ověření podpisu, na nějž jsou reference v CAdES-C. (Zatímco CAdES-C obsahoval jen reference, tak CAdES-X Long obsahuje celá data).
- CAdES-X Type 1, který pomocí časového razítka kompletně zajišťuje CAdES-C.
- CAdES-X Type 2, který pomocí časového razítka zajišťuje informací nutných pro ověření podpisu uvedené v CAdES-C (nikoliv celý CAdES-C).
- CAdES-X Long Type 1 a Type 2 obsahují jak kompletní data, tak i časové razítko.
- CAdES-A, který obsahuje archivní časové razítko. Po čase může být vloženo další časové razítko atd.

10.2 Elektronická obálka (EnvelopedData)

Elektronická obálka obsahuje šifrovaný obsah a pro každého adresáta pak šifrovaný klíč, kterým byl šifrován obsah zprávy.

Princip elektronické obálky spočívá v tom, že zpráva (vstupní data) se šifruje náhodně generovaným symetrickým klíčem (content-encryption key nebo někdy ne zcela správně označovaným session key). Každý adresát zprávy má pak ve zprávě svůj výskyt položky „Informace pro adresáta“ (RecipientInfo). Ta obsahuje náhodně generovaný symetrický klíč (content-encryption key). Jenže jej neobsahuje jen tak, ale v zašifrované podobě, aby se k němu dostal pouze adresát, pro kterého je určena konkrétní struktura RecipientInfo (Obrázek 10.9).

Zpráva EnvelopedData má následující strukturu:

- Položka Verze obsahuje verzi formátu zprávy.

- Nepovinná položka „Informace o odesílateli“ je určena pro odesílatelovy certifikáty a CRL. Oceníme je také v případě, kdy využíváme kryptografii založenou na principu výměny klíčů, tj. v případě Diffie-Hellman metody.
- Položka RecipientInfos obsahuje pro každého adresáta (nebo skupinu adresátů, jakou je např. konference) jeden výskyt struktury RecipientInfo, tj. informaci pro adresáta obsahující buď zabezpečený šifrovací klíč zprávy, nebo alespoň jeho identifikaci.
- Položka „Šifrovaná zpráva“ obsahuje vlastní zašifrovanou zprávu.
- Položka „Nezabezpečené atributy“ je obdobou nepodepisovaných atributů v případě digitálně podepsané zprávy:

11 XML a elektronický podpis

Pokud budeme studovat standardy CAdES a XAdES, tak zjistíme, že veškerá jejich logika je skryta v upřesnění a přidání dalších tzv. podepisovaných a zejména tzv. nepodepisovaných atributů elektronického podpisu. Rozdíl mezi podepisovanými a nepodepisovanými atributy spočívá v tom, že podepisované atributy se zahrnují do výpočtu otisku, jež vstupuje do výpočtu elektronického podpisu. Naopak nepodepisované atributy se do výpočtu elektronického podpisu nezahrnují, a tak je možné je kdykoliv k podpisu připojit (bohužel i z podpisu odebrat). Zajímavostí je, že nepodepisované atributy mohou obsahovat např. časové razítko nebo kontrasignaturu, které samy obsahují elektronický podpis. Jedná se ale o jiný (nezávislý) elektronický podpis.

XAdES vychází se standardu DSIG jehož architektura je znázorněna na obr. Obrázek 11.1.

DSIG vkládá do XML dokumentu podpis (element `Signature`). Podepisuje se ta část dokumentu, která je specifikována („rerferencována“) elementem Reference. Myšlenkou bylo specifikovat původní XML dokument (před vložením podpisu).

Zajímavé je, že nikde není psáno, že element `Signature` by měl být v dokumentu jen jeden, tj. do dokumentu můžeme vkládat více podpisů. V dalších podpisech můžeme referencovat jinou část dokumentu (včetně jiného podpisu). Nemusí se proto hrát na paralelní a sériové podpisy. XAdES přesto zavádí i element Kontrasignatura, tak jak ji poznáme u formátu CAdES.

Podpis DSIG (element `Signature`) obsahuje následující elementy (Obrázek 11.1):

- `SignedInfo` specifikující co se podepisuje.

  Tento element obsahuje následující elementy: `CanonicalizationMethod` – kanonizační algoritmus (např. algoritmus C14n), který byl aplikován na element `SignedInfo` před tím, než byla počítána hodnota podpisu. Cílem je zajistit jednoznačnou reprezentaci zápisu XML (vypustit měkké mezery apod.). `SignatureMethod` – algoritmus pro výpočet otisk.

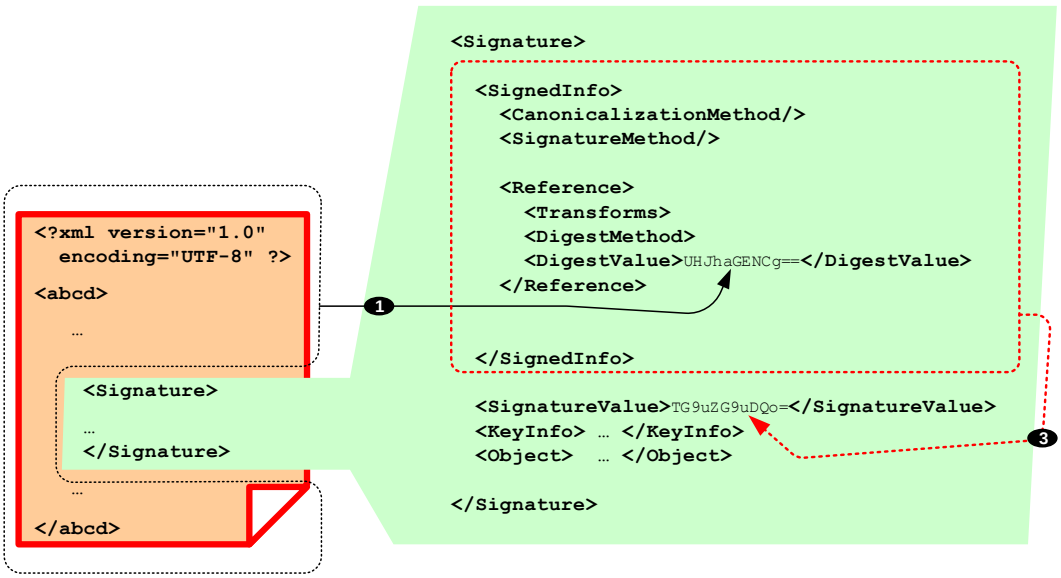
- Jeden nebo více elementů `Reference` specifikují části XML dokumentu, které mají být zahrnuty do výpočtu podpisu.

  Element `Reference` obsahuje následující elementy: `Transforms` – specifikující transformaci, která byla na XML dokument aplikována před výpočtem podpisu – např. XPath řetězec. Algoritmů může být uvedeno i více. Např. jako první může být uveden řetězec XPath a jako druhý algoritmus pro kanonizaci, která se provede na XML text před výpočtem otisk (`DigestValue`). `DigestMetod` – specifikuje algoritmu výpočtu otisk, `DigestValue` – specifikuje Base64 kódovanou hodnotu otisk.

- Element `SignatureValue` obsahuje Base64 kódovanou hodnotu podpisu.

- Element `KeyInfo` obsahuje klíče pro verifikaci podpisu. Např. certifikáty veřejného klíče (X.509).

- Volitelný element `Object` se později stal základem XAdES. DSIG, ale nedefinuje podepisované a nepodepisované elementy (vlastnosti) podpisu. Je zde v podstatě jako vrátka pro budoucí využití.



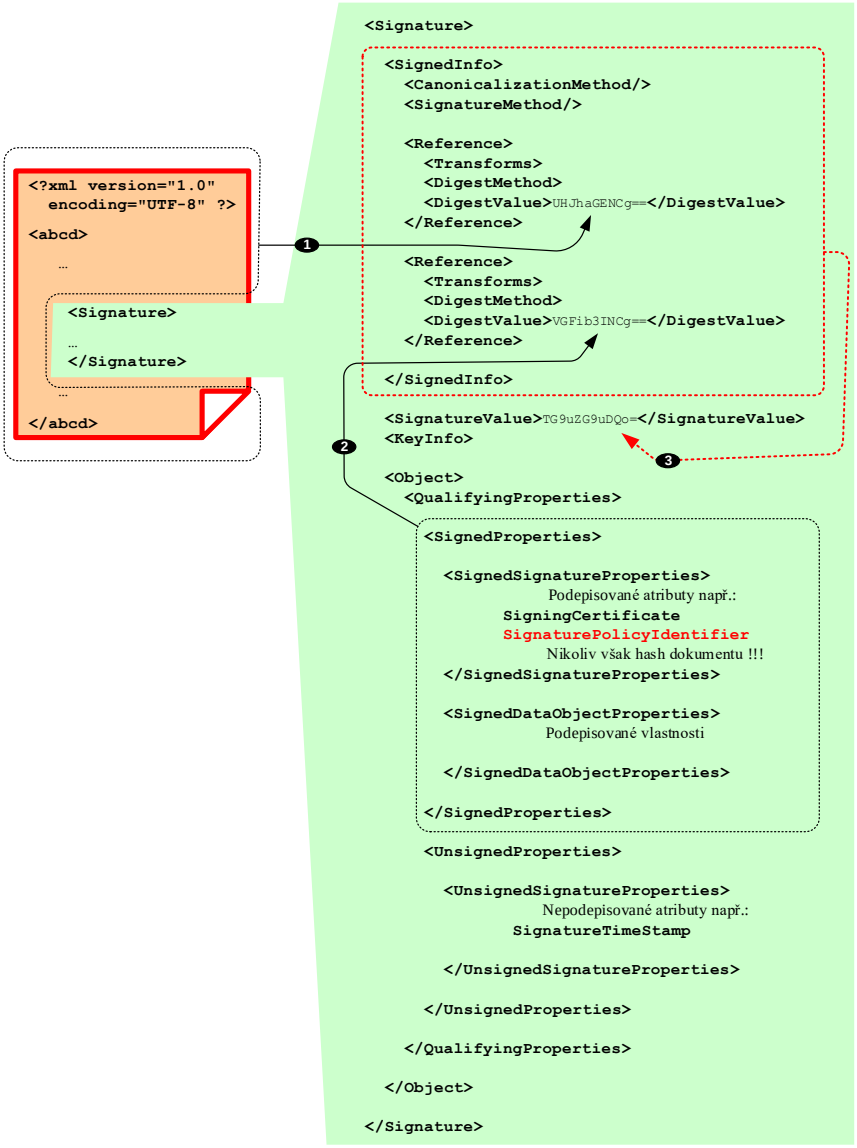
Obrázek 11.1 Architektura formátu DSIG

11.1 XAdES

XAdES *(XML Advanced Electronic Signatures)* využívá element Object, aby do jeho elementu `QualifyingProperties` vložil podepisované a nepodepisované elementy (obdoba podepisovaných a nepodepisovaných atributů formátu CAdES). Tyto elementy se nazývají *Properties*, česky asi vlastnosti. Jedinou zvláštnosti je, že podepisované (teoreticky i nepodepisované) vlastnosti se dělí na:

- `SignatureProperties` – týkající se podepisovaného textu.
- `DataObjectProperties` – týkající se datových objektů podpisu.

Architektura formátu XAdES je na obr. Obrázek 11.2. Podpis XAdES obsahuje dva elementy `Reference`: první obsahuje otisk z podepisovaného XML dokumentu s druhou z podepisovaných vlastností (obsahují podepisované elementy). Výsledný podpis je soukromým klíčem šifrovaný otisk z elementu `SignerInfo` a nakonec kódovaný Base64.



Obrázek 11.2 Architektura XAdES

Formát XAdES zavádí formáty XAdES-BES, XAdES-EPES, XAdES-T, XAdES-X, XAdES-X a XAdES-C. Jejich význam je obdobný jako v případě CAdES.

Jak jsem se již zmínil, tak základem formátu XAdES je DSIG. DSIG je podstatně jednodušší formát, který si nehraje na podepisované a nepodepisované elementy. Pakliže použijeme čistý DSIG, tak nemáme kam vložit časové razítko podpisu (o identifikátoru podepisovacího certifikátu ani nemluvě), proto ve všech případech, kdy chceme použít XML Signature

pro zaručený podpis bychom měli využít alespoň ten nejjednodušší XAdES-BES, protože z formátu DSIG formát XAdES už nikdy neuděláme.

12 PDF a elektronický podpis

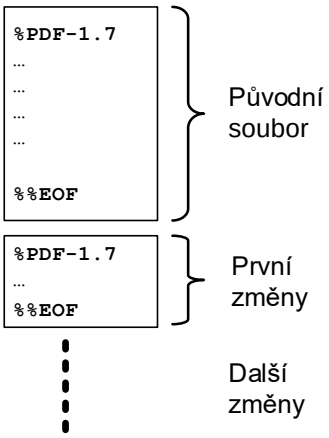
PDF (Portable Document Format) je standard formátu dokumentů vytvořený firmou Adobe. Firma Adobe tento standard otevřela a prosazuje jej jako mezinárodní normu. Aktuální verze PDF 1.7 byla v roce 2008 přijata ISO jako standard ISO 32000-1. Již řadu let se očekává vydání PDF verze 2.0, která by měla vyjít jako standart ISO 32000-2.

V případě listinných dokumentů si pod slovem dokument představíme finální verzi listiny, která se již nemění. Pokud na listinu např. rukou připíšeme poznámku, pak je to listina s jasně význačnou poznámkou. Pokud bychom chtěli na listinu něco připsat tak, aby to nebylo zřetelně odlišeno od původního textu, pak se jedná o pozměňování listiny.

V případě elektronických dokumentů si přestavíme dokument vytvořený textovým editorem, který je možné znovu a znovu otevírat a modifikovat. Finální verzi dokumentu pak zpravidla vytiskneme na tiskárně a chováme se k ní jako k listině.

PDF je formát finálních elektronických dokumentů. PDF soubor již neotevíráme v klasickém editoru a neměníme jej. Avšak můžeme jej rozšířit o poznámky, vyplnit pole formuláře atp. Tj. vyměnit nebo zrušit některé objekty dokumentu.

Vždy ale původní informace v dokumentu zůstanou



Obrázek 12.1 Změny v PDF souboru

a změny se připojí na konec PDF souboru ve tvaru jakoby nového PDF souboru, ale obsahující jen modifikované objekty nebo indikaci zrušených objektů (viz Obrázek 12.1).

Je pochopitelně možné i PDF soubor otevřít textovým editorem (např. Notepad) a pozměňovat jej. Je si třeba ale uvědomit, že např. vložit odstavec do stránky s tím, že se obsah na všech dalších stránkách o odstavec posune, znamená, že musíme pracně pozměnit nejenom aktuální stránku, ale i všechny následující. Dokonalou ochranou proti pozměnění PDF souboru je jeho elektronický podpis.

12.1 PDF soubor

Otevřeme-li si PDF soubor v jednoduchém editoru (např. Notepad), pak uvidíme, že PDF soubor se skládá z několika částí (Obrázek 12.1):

- Záhlaví specifikující verzi PDF standardu, ve kterém byl soubor vytvořen.
- Tělo obsahující jednotlivé objekty, které jsou sekvenčně uspořádány.
- Křížové reference. Každý objekt (těla) má zde jeden řádek, který obsahuje deseticiferný posun objektu od počátku PDF souboru, pěticifernou generaci (verzi) objektu a příznak je-li objekt funkční (n) nebo zrušen (f).
- Zápatí, které zejména obsahuje označení který objekt je kořenovým (/Root) objektem PDF souboru a kolik položek obsahují křížové reference (/Size). Kořenový objekt je typu /Catalog (na Obrázek 12.3 je to objekt číslo 1 na který je v zápatí položce /Root odkaz „1 0 R“).

12.2 Objekt

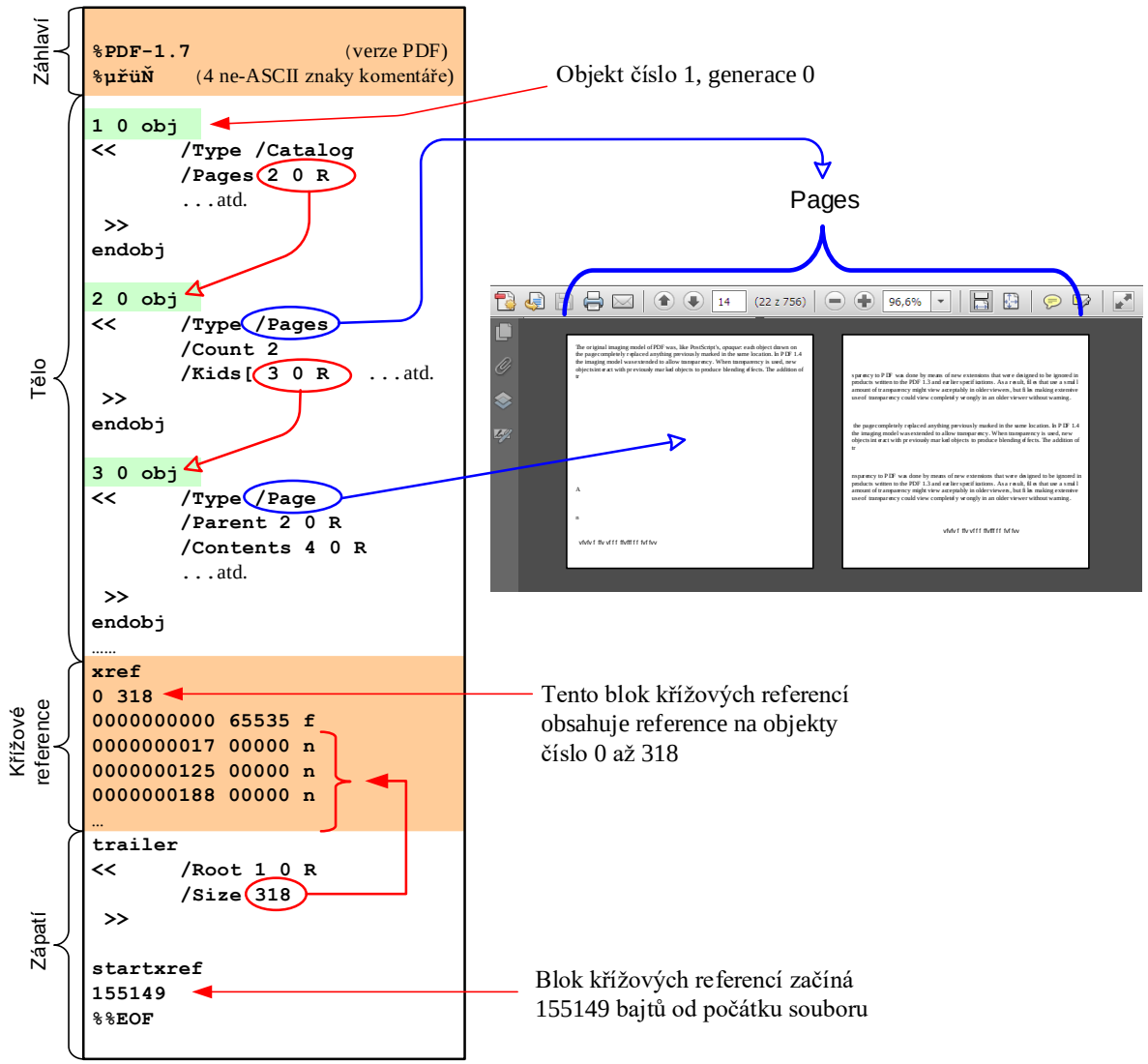
Tělo PDF souboru obsahuje objekty. Každý objekt začíná řádkem obsahujícím číslo objektu, generaci (verzi) objektu a klíčové slovo obj. Např. všechny objekty na Obrázek 12.2 mají generaci nula. Pouze na počátku křížových referencí vidíme odkaz na objekt 0 s o generaci 65535, který je zrušen. Jedná se o atypický objekt číslo 0 specifikující celý soubor a nevyskytující se v těle.

Za řádkem s klíčovým slovem obj zpravidla následuje asociativní pole uzavřené ve dvojitých špičatých závorkách << >>. Asociativní pole obsahuje dvojce klíč a hodnota. Např. klíč /Count s hodnotou 2 nám v objektu číslo 2 sděluje, že PDF soubor má dvě stránky. Za asociativním polem někdy následuje tok dat uzavřený závorkami stream a endstream.

V objektech se můžeme nepřímo odvolávat na jiné objekty tak, že uvedeme číslo objektu, jeho generaci a písmeno R (reference). Např. objekt číslo 1 v klíči /Pages obsahuje nepřímou referenci na objekt číslo 2.

12.3 Podpis v PDF dokumentu

Jelikož PDF dokument se více či méně tváří jako listinný dokument, tak v PDF je podporován jak elektronický podpis, tak i viditelný podpis, které mohou být hypertextově provázány.



Obrázek 12.2 Příklad struktury PDF souboru

Elektronický podpis je v PDF chápán obecně. Sám standard PDF 1.7 předpokládá dva typy podpisů:

- Podpis na bázi veřejného/soukromého klíče (PKI podpis). V tomto případě se využívá:
  - Standard CMS („PKCS#7“) pro podepisování dokumentů s tím, že struktura CMS podpisu může obsahovat jen jeden podpis, tj. nelze k tomuto podpisu přidat další paralelní podpis.
  - Standard XAdES (podpis XML struktur) pro podpis částí PDF dokumentu, které mají strukturu XML. Zejména se jedná o XML strukturu vyplněných hodnot PDF formulářů (tzv. XFA formulář).
- Biometrický podpis.

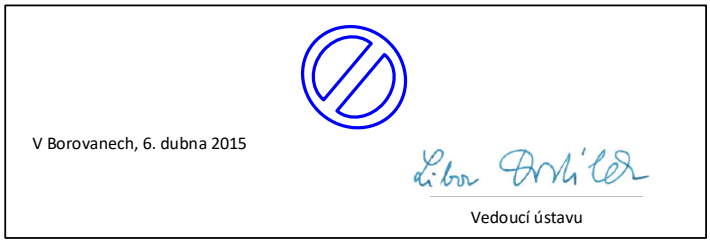
12.3.1 Viditelný podpis

Elektronický podpis má pro laického uživatele tu nepříjemnou vlastnost, že je skryt v bitech souboru – není viditelný. Např. programy elektronické pošty u podepsaných zpráv zobrazují skutečnost, že zpráva byla elektronicky podepsána zpravidla ikonou pečeti někde v horní roku okna zobrazené zprávy. Neznalý uživatel často ani neví, co tato ikona znamená.

U listinných dokumentů jsme přitom zvyklí na spodní díl listiny (Obrázek 12.3).

Aby se elektronická forma přiblížila listinné, pak je i v elektronických dokumentech vyžadován obdélník, který obsahuje obdobné informace. Včetně např. faksimile rukou psaného podpisu. Mluvíme pak o dvou podpisech:

- O elektronickém podpisu (např. PAdES)
- O viditelném podpisu, který napodobuje listinný podpis a jeho okolí (připomíná Obrázek 12.3).



Obrázek 12.3 Tradiční podpis na listinách

V elektronickém dokumentu se jedná o jeho dvě různé části, které mohou provázány např. tím, že kliknutím myši na viditelný podpis se nám zobrazí informace o elektronickém podpisu. Příklad je uveden na Obrázek 12.4. Po otevření tohoto dokumentu se mi nejenom při horním okraji zobrazil bledě modrý pruh signalizující, že někde uvnitř dokumentu se tajemně skrývá elektronický podpis, ale navíc se zobrazil obdélník s viditelným podpisem. Na tento obdélník je pak navázaná akce zobrazení vlastností podpisu. Což se objevilo po kliknutí na obdélník.

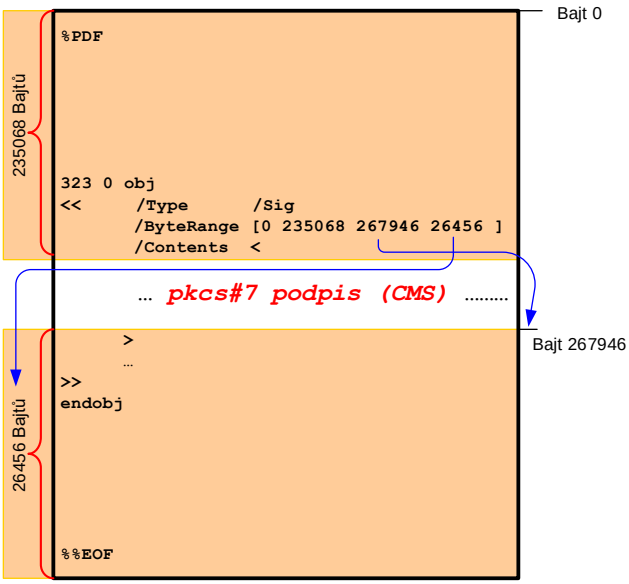
U dokumentů nahrazujících listinné dokumenty bychom vždy měli dbát, aby kromě elektronického podpisu obsahovaly i viditelný podpis. Důvod je velice prostý. Dokumenty mohou být archivovány po dobu desítek nebo stovek let. V této době se může stát, že např. formát PDF nebude již dále podporován a archiv bude nucen ke konverzi formátu. Při konverzi formátu dochází k znehodnocení elektronického podpisu. Viditelný podpis pak může být jedinou indicií, že dokument byl podepsán.

Obsah viditelného podpisu specifikuje standard ETSI TS 102 778-6. Jsou v něm zajímavé postřehy, jako např. že viditelný podpis by neměl obsahovat jiné informace o podepisující osobě než ty, které lze ověřit pomocí certifikátu atp.

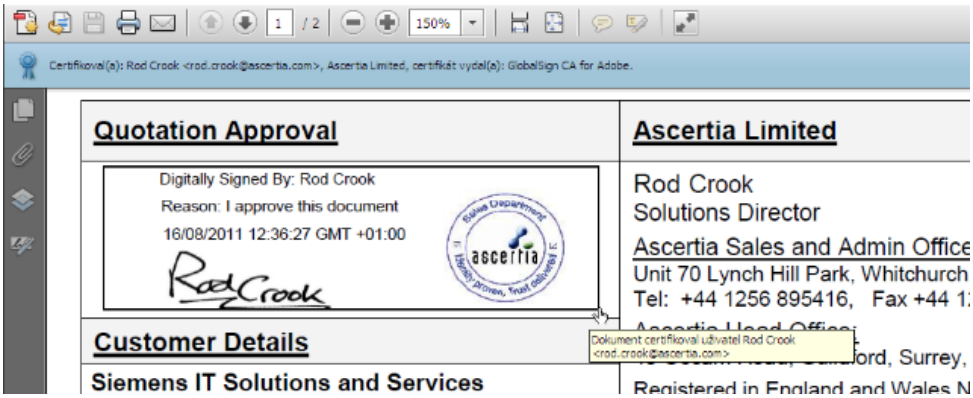
podpisem, který je provázán s obdélníkem obsahujícím viditelný podpis

12.3.2 Elektronický podpis v PDF dokumentu

Elektronický podpis se vkládá do objektu typu /Sig. Jelikož se jedná o interní podpis, tak se skrývá někde uvnitř PDF dokumentu. Jenže do výpočtu podpisu se samotný podpis nemůže vstoupit (musí se přeskočit). Co se do výpočtu podpisu započítává je specifikováno klíčem /ByteRange. Jeho parametrem je pole (pole se vždy v PDF uvádí v hranatých závorkách [ ]) specifikující části souboru, které se zahrnují do výpočtu podpisu. Každá část je tvořena dvojicí: ofset od počátku souboru a délka části (Obrázek 12.5).



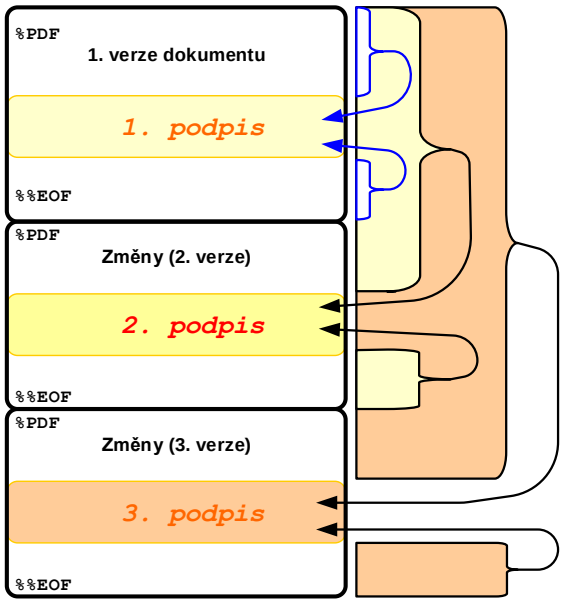
Obrázek 12.5 Do výpočtu se zahrnují části PDF dokumentu určené /ByteRange



Obrázek 12.4 Fragment faktury s elektronickým

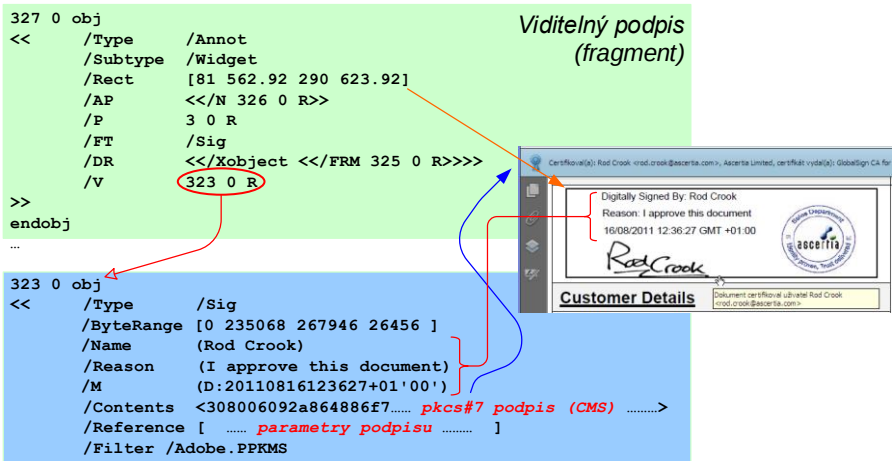
12.3.3 Vícenásobný podpis v PDF

Zatímco standard CMS umožňuje paralelní i sériové podpisy, v případě PDF se vkládá do CMS jeden podpis. PDF neumožňuje paralelní podpisy, ale umožňuje sériové podpisy.



Obrázek 12.6 PDF umožňuje jen sériové podpisy

PDF však pro sériové podpisy nevyužívá nepraktický, nepodepisovaný atribut kontrasignatura známy z CMS, ale používá vlastní řešení, které je bližší reálnému životu. Prakticky totiž vznikne 1. verze dokumentu, která se podepíše prvním podpisem. Další uživatel může připojit k dokumentu připomínky (změny) a připojit podpis pod původní podepsaný dokument včetně změn (Obrázek 12.6). Výsledkem je, že se vytváří několik sériových podpisů – každý je tvořenou samostatným CMS podpisem.



Obrázek 12.7 Viditelný podpis svázaný s PKI podpisem

12.3.4 PKI Podpis

Jak již bylo zmíněno, tak PDF využívá CMS podpis. Samotný podpis CMS podpis je obsahem položky o klíči /Contents (Obrázek 12.5). Na obrázku je uveden v jednoduchých špičatých závorkách, tj. každý bajt je zobrazen pomocí dvou šestnáctkových číslic.

S délkou CMS podpisu je spojena zajímavost, která asi každého překvapí, když se podívá do zdrojového textu PDF dokumentu na elektronický popis (stačí PDF dokument otevřít nějakým jednoduchým textovým editorem jako je Notepad). Překvapí ho totiž velké množství znaků 00 (šestnáctkově) na konci podpisu.

Důvod je prostý. Pokud se počítá s tím, že elektronický podpis bude později rozšířen např. o certifikáty, revokační informace či časová razítka, tak může dojít ke zvětšení velikosti podpisu. Avšak PDF formát změny dokumentu bere jako další verzi, které ukládá na konec souboru (za %%PDF). Jenže CMS podpis je uzavřená datová struktura, kterou nelze roztrhnout na dvě části. Řešením je proto nejprve vložení takového množství znaků 00 (šestnáctkově) na místo podpisu, kolik by zabrala maximální prakticky možná varianta podpisu. Následně se tyto znaky přepisují rozšiřovanou strukturou elektronického podpisu. Množství nadbytečných znaků 00 se tím jen zmenší.

12.3.5 XML podpis v PDF

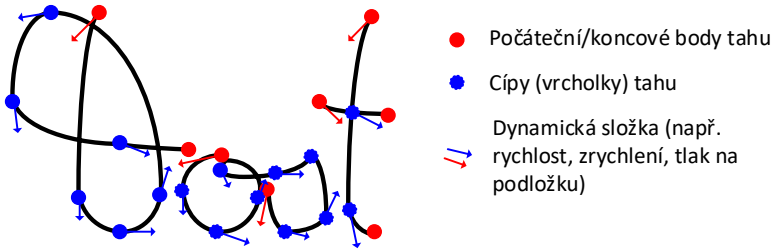
Součástí PDF dokumentu mohou být i části v XML. Jedná se např. o vyplněné hodnoty formulářových polí (tzv. XFA struktura). PDF nic nebrání tomu, aby tyto části byly elektronicky podepsány pomocí XML prostředků pro elektronický podpis.

PDF formulář může být podepsán, proto vyplněná pole nelze ukládat přímo dovnitř tohoto formuláře (porušila by se integrita a podpis by ztratil smysl). Vyplněná pole se ukládají na konec formuláře do vloženého XML dokumentu (obdobným způsobem jako vyplněná pole v HTML formuláři).

Tj. pokud by měl klient podepsat vyplněný formulář, tak musí podepsat jen tu vloženou XML strukturu – podepsat jen hodnoty, které vyplnil. Jelikož se jedná o XLM strukturu, tak použije XML podpis, např. některý z podpisů typu XAdES. Tento typ podpisu specifikuje norma ETSI TS 102 778-5.

12.3.6 Biometrický podpis

Pokud zkoumáme tah podpisu, pak si na něm nalezneme charakteristické body (Obrázek 12.8). V případě statického podpisu nás zajímá jen poloha bodu a jedná-li se o počáteční bod obloučku, koncový bod obloučku nebo cíp obloučku.



Obrázek 12.8 Charakteristické body tahu podpisu

V případě dynamického podpisu se pak navíc ještě snímají dynamické parametry jednotlivých charakteristických bodů. Zajímavými dynamickými parametry jsou např. rychlost tahu, zrychlení, tlak na podložku apod.

Pro elektronické vyhodnocování biometrických podpisů vytvoří uživatel více (zpravidla nejméně tři) podpisové vzory. Následný podpis se pak vyhodnocuje proti podpisovým vzorům.

Dynamický biometrický podpis není žádná horká novinka. To lze doložit tím, že od roku 2007 pro něj máme ISO normu ISO/IEC 19794 -7 Biometric Data Interchange Formats – Part 7: Signature/Sign time series data. Ve stádiu draftu je pak standard ISO/IEC DIS 19794-11 Biometric data interchange formats -- Part 11: Signature/sign processed dynamic data. Tato norma je např. využívána cestovními doklady s biometrickými prvky.

12.3.6.1 Elektronický biometrický podpis

Slovem „Elektronický podpis“ nařízení eIDAS rozumí: „data v elektronické podobě, která jsou připojena k jiným datům v elektronické podobě nebo jsou s nimi logicky spojena, a která podepisující osoba používá k podepsání“.

Problémem je to „logické spojení“ s datovou zprávou. Prakticky se to provede tak, že se ze zprávy spočte otisk a vytvoří se datová struktura, která obsahuje spočtený otisk a dynamický biometrický podpis. Jelikož dynamický biometrický podpis je z bezpečnostního hlediska obdobou soukromého klíče, tj. jedná se o bezpečnostně citlivý údaj, tak se výsledná struktura vloží do elektronické obálky (zašifruje se).

Vyzrazení dynamického biometrického podpisu je obdobně nebezpečné jako vyzrazení soukromého klíče. Při vyzrazení soukromého klíče se útočník v elektronickém světě může vydávat za původního držitele klíče. Obdobně při vyzrazení dynamického biometrického podpisu může útočník tento podpis podvrhovat. Tj. libovolně kopírovat do falšovaných dokumentů. Leze se podvržení bránit např. tím, že pro různé účely budu k podpisu připojovat ještě např. číslice nebo jednoduché obrázky.

12.3.6.2 Hanler a SubFilter

PDF složitější akce provádí pomocí specializovaných modulů – handlerů. Někdy se jim také říká filtry, protože se na ně v PDF jazyce odkazuje pomocí klíče /Filter.

Pro oblast kryptografie se používají kryptografické handlery. Standardem se nepřímo staly handlery vytvořené firmou Adobe.

Handler	
Standard (vestavěný handler)	Standardní bezpečnostní handler pro šifrování heslem
Adobe.PubSec	Handler elektronické obálky CMS (PKCS#7)
Adobe.PPKLite	Handler elektronického podpisu CMS (PKCS#7)
Entrust.PPKEF	Příklad firemního handleru firmy Entrust pro PKI
SOFTPRO#20DigSig#20Security	Příklad firemního handleru firmy Softpro pro dynamický biometrický podpis

Nevýhodou handleru je, že se jedná o firemní software. Pokud použijeme konkrétní handler při vytvoření dokumentu, tak bychom stejný handler měli použít i pro čtení (verifikaci) dokumentu. Nevýhodou je tedy potencionální nekompatibilita mezi různými handlery.

Jinou cestou je nespecifikovat software, který byl použit při vytvoření dokumentu, ale formát dat, která byla handlerem vytvořena. Tento formát dat se specifikuje pomocí klíče /SubFilter. Jsme pak v

opačné situaci: hledáme software, který podporuje příslušný SubFilter.

SubFilter	
adbe.pkcs7.s5	Elektronická obálka PKCS#7 (eventualita s5)
adbe.x509.rsa_sha1	Elektronický podpis PKCS#7
adbe.pkcs7.detached	Externí elektronický podpis PKCS#7
ETSI.CAdES.detached	PAdES-BES nebo PAdES-EPES
ETSI.RFC3161	Časové razítko, PAdES-LTV (takto vytvořená časová razítka proto software zpravidla zobrazuje jako další podpis)

Speciálně v případě dynamických biometrických podpisů se často klade otázka, zdali takový podpis vytvořený pomocí software jednoho dodavatele je kompatibilní se software jiného dodavatele. Aby tomu tak bylo, pak by musel existovat nějaký standardizovaný (alespoň všeobecně rozšířený) SubFilter. Já jsem se ale s takovým nesetkal.

V tabulce příkladů SubFilter si všimněte, že pro vytváření podpisů typu PAdES volíte SubFilter obsahující řetězec CAdES. To je také vysvětlení, proč, když v Adobe PRO chceme vytvořit podpis typu PAdES, tak zvolíme „Výchozí formát podepisovaného dokumentu“ CAdES.

12.3.6.3 Elektronický biometrický podpis v PDF

Různí dodavatelé dodávají řešení pro vytváření dynamického biometrického podpisu PDF dokumentů. Princip je znázorněn na obr. Obrázek 12.9. Na tabletu se vytvoří rukou psaný podpis. Tablet jej online nasnímá a vytvoří:

- Statický podpis
- Dynamický podpis

Statický podpis se doplní o další údaje (např. čas podpisu, důvod podpisu apod.) a vloží se do dokumentu jako viditelný podpis.

Dynamický podpis se sřetězí otisk dokumentu a vznikne základ pro dynamický elektronický podpis. Jelikož se jedná o bezpečnostně citlivý údaj, tak se před vložením do dokumentu vloží do elektronické obálky (zašifruje se). Šifrovat se může jak tajným klíčem, tak veřejným klíčem.

Prakticky zajímavá je možnost, kdy si na specializovaném pracovišti vygenerujeme dvojici veřejný a soukromý klíč. Veřejný klíč certifikujeme. Certifikát tohoto veřejného klíče používáme k vytvoření elektronické obálky dynamických biometrických podpisů vytvářených ve firmě/úřadu. Certifikát pak distribuujeme na všechny počítače, kde se bude vytvářet dynamický biometrický podpis.

Pokud chceme dynamický biometrický podpis verifikovat. Pak se takto podepsaným dokumentem dostavíme na specializované pracoviště disponující příslušným soukromým klíčem. Pracoviště dešifruje dynamický elektronický podpis, čímž získáme dynamický podpis. Dynamický podpis pak můžeme verifikovat proti podpisovým vzorům uloženým v databázi nebo se podepisovaná osoba dostaví na pracoviště a prokáže, že podpis opravdu vytvořila.

12.3.6.4 Jak dynamický biometrický podpis v PDF uložen?

Je uložen zcela obdobně jako elektronické podpisy založené na PDF. Tj. opět se využije objekt typu /Sig, rozsah podepisované části souboru se určí pomocí klíče /ByteRange a vlastní podpis se uloží pod klíčem /Contents.

Obdobně jako elektronický podpis založený na PKI, může být i takto vytvořený dynamický biometrický podpis hypertextově provázán s viditelným podpisem.

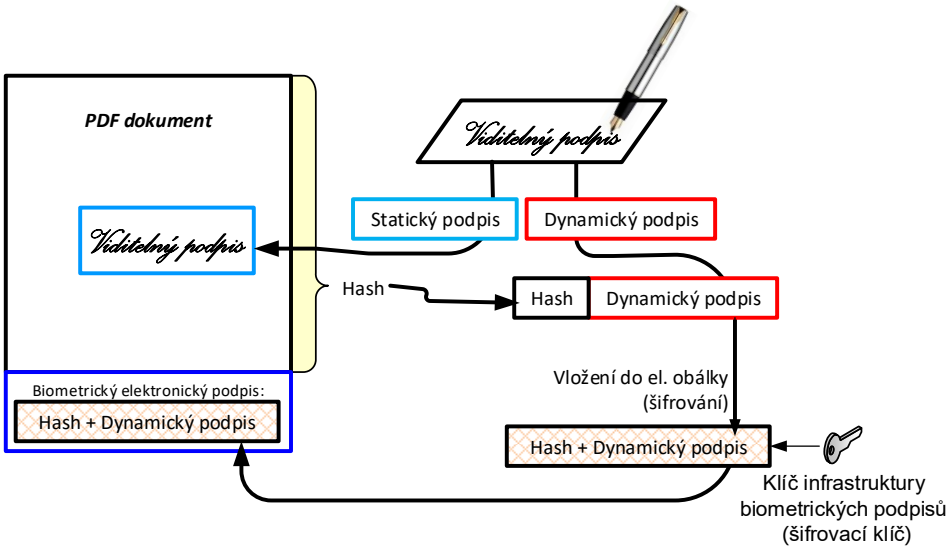
12.3.6.5 Jak dynamický biometrický podpis využít v praxi?

Klasické použití bude zejména pro případy, kdy nám jde o podpis jako o rituál a nelpíme tolik na otázce, jestli opravdu bezezbytku nahrazuje rukou psaný podpis. Takové použití je např. v pojišťovnictví. Kdy klient sice podepisuje pojistnou smlouvu, ale systém může být nastaven tak, že finálně autorizuje až první platbou. Obdobně při likvidaci pojistné události je podpis poškozeného opět spíše rituálem.

V těchto případech se zpravidla vytváří podpisy dva:

- Podpis pojistníka (resp. poškozeného)
- Podpis pracovníka pojišťovny.

Jenže pracovník pojišťovny zpravidla může vytvářet podpis na bázi PKI (např. je vybaven příslušnou čipovou kartou). Takže jako první podpis se provede dynamický biometrický podpis (pojistníka/poškozeného) a jako druhý pak PKI podpis pracovníka (obr. Obrázek 12.10), který může být např. typu PAdES-BES a následně se může doplnit časovým razítkem (až pracovník dorazí na svou pobočku).



Obrázek 12.9 Připojení dynamického biometrického podpisu k PDF dokumentu

12.4 Kombinace podpisů

Jednotlivé podpisy je možné i kombinovat. Je třeba si ale uvědomit, že pokud se podepíše dokument i s nějakým jeho vnitřními podpisy, tak se tyto vnitřní podpisy zakonzervují vnějším podpisem. Dlouhodobost podpisu má pak cenu ošetřovat zejména u vnějšího podpisu.

12.5 Zabezpečení PDF

PDF nám umožňuje:

- Nastavovat přístupová oprávnění.
- Šifrovat dokument.
- Podepisovat dokument.

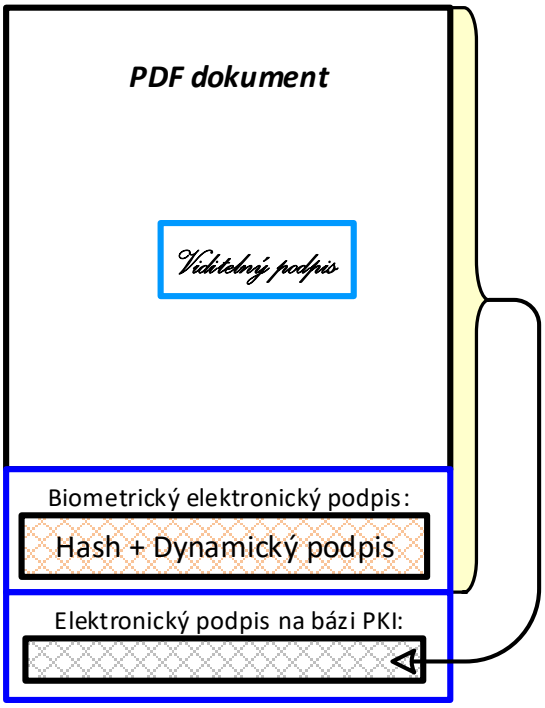
12.5.1 Nastavování přístupových oprávnění

Cílem přístupových oprávnění v PDF je omezit čtenáři přístup k obsahu dokumentu a některé činnost s dokumentem. Jenže pokud software obsah dokumentu dešifruje, tak je již na tvůrci software, zdali nastavená oprávnění respektuje nebo nikoliv. Lze se tak setkat se software, který cíleně „odemkne“ i činnost původně zakázané.

Asi trochu překvapivé je, že přístupová oprávnění (včetně uživatelský rolí) se spojují se šifrováním nebo elektronickým podpisem. Přitom oprávnění i uživatelské role spojené se šifrování a spojené s elektronickým podpisem jsou odlišné.

12.5.2 Šifrování dokumentu

Dokument může být šifrován heslem (handler Standard) nebo vložen do CMS (PKCS#7) elektronické



Obrázek 12.10 Biometrický elektronický podpis se často doplňuje o podpis na bázi PKI

obálky za využití PKI (např. SubFilter adbe.pkcs7.s5) .

Vestavěný handler Standard používá 32 znaků dlouhé heslo pro šifrování. Pakliže uživatel zadá heslo kratší, pak se doplní znaky řetězce < 28 BF 4E 5E 4E 75 8A 41 64 00 4E 56 FF FA 01 08 2E 2E 00 B6 D0 68 3E 80 2F 0C A9 FE 64 53 69 7A > na 32 bajtů. Hypoteticky se tak můžeme setkat i se souborem zašifrovaným prázdným heslem. V takovém případě by se použil jako heslo uvedený řetězec.

Standardní handler umožňuje zřídit dvě uživatelské role: vlastník a uživatel. Každá role může mít své heslo a mohou pro ni být nastavena jiná přístupoví oprávnění. Přístupovými oprávněními jsou např.:

- Před otevřením souboru vyžádat heslo (uživatele/vlastníka).
- Modifikace obsahu dokumentu.
- Kopírování obsahu dokumentu.
- Přidávání a modifikace poznámek.
- Tisk dokumentu.
- Vyžádání hesla pro tisk.
- Tisk jen v určité kvalitě.
- Vyplňování formulářů.
- Vkládání, otáčení nebo rušení stránek.

Pokud nepoužijeme standardní handler, ale SubFilter začínající na adbe.pkcs7.s, pak podobně jako v elektronické poště, můžeme mít více uživatelů („adresátů“). Pro každého z nich je pak jeho veřejným klíčem šifrován nejenom náhodný tajný šifrovací klíč dokumentu, ale řetězec jeho přístupových práv, která jsou obdobní jako v případě podpisu dokumentu.

12.5.3 Podpis dokumentu

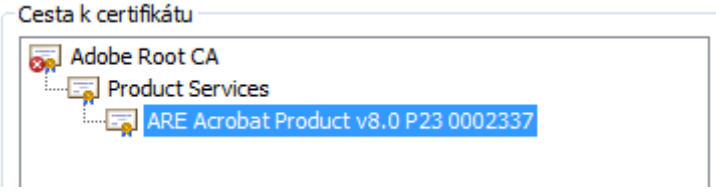
Nyní nás nebude zajímat, jestli je podpis na bázi PKI nebo biometrie, ale z hlediska funkčnosti. Z tohoto pohledu PDF mj. rozlišuje následující typy podpisů:

- Běžný podpis. Někdy nazývaný schvalovacím podpisem. Tento podpis jsme měli v předchozích odstavcích na mysli.
- Certifikační podpis. Může být v dokumentu nejvýše jeden a to vždy jako první podpis dokumentu. Jeho význam lze pochopit u formulářů, kdy autor formuláře tímto podpisem stvrzuje pravost formuláře. Často se stává, že uživatel nepodepisuje formulář, ale běžný dokument. Při prvním podpisu software nabízí možnost podepsat nebo certifikovat, tj. jestli se má vytvořit běžný podpis nebo certifikační podpis. Mnozí uživatelé sice chtějí vytvořit běžný podpis, ale vytvoří certifikační podpis. Nemusí to být na závadu, ale software nám zobrazí, že dokument byl „certifikován“ místo podepsán. Přitom slovo „certifikován“ nemá žádnou souvislost s certifikátem veřejného klíče. Certifikační podpis nemusí podepsat úplně celý dokument, ale jen jeho část (viz dále)
- Nejvýše dva podpisy uživatelských práv.

Certifikační podpis je vždy spojen s objektem DocMDP (Modification Detection and Prevention), kde autor dokumentu definuje, jestli se jedná o finální dokument (celý dokument je podepsán) nebo některé jeho části je možné modifikovat (nejsou zahrnuty do podpisu).

Podpis uživatelských práv je asi to nejzajímavější a nejtajuplnější věc na celém formátu PDF. Tyto podpisy byť obsažené v PDF dokumentu se v Acrobatu nezobrazí, jsou skryté. Mají v podstatě dva významy:

- Přístupová práva jako např.: OnLine vkládání poznámek, export souboru, uložení vyplněného formuláře, import dat atd.
- Obchodní model firmy Adobe. Tato firma poskytuje zdarma Acrobat Reader. Pokud chceme, aby programu Acrobat Reader bylo možné např. podepisovat dokument, pak jej musíme opatřit elektronickým podpisem uživatelských práv, kde je uživatelským právem možnost vložit elektronický podpis. Toho lze dosáhnout např. v Adobe Pro volbou „Uložit jako“ -> „Uložení s rozšířením pro Reader“. Tato volba uloží do PDF dokumentu skrytý elektronický podpis práv zvolených uživatelských práv, z nichž jedno může být možnost podepisování dokumentu programem Acrobat Reader. Tj. Acrobat Pro je distribuován s dvojící veřejný a soukromý klíč a veřejný klíč je certifikován dle X.509. Např. můj instalovaný Adobe Pro vytvořil podpis uvedený na obr. Obrázek 12.11. Po otevření takto podepsaného dokumentu v programu Adobe Reader se nám objeví možnost Rozšíření, kde nalezneme i tlačítko pro vytvoření elektronického podpisu. I když na mém počítači se certifikační autorita Adobe Root CA jevila jako nedůvěryhodná, tak můj Adobe Reader byl spokojený a podpisoval.



Obrázek 12.11 certifikát vložený do mé distribuce Adobe Pro

12.6 PDF/A

PDF/A neoznačuje konkrétní datový formát, ale rodinu formátů. Myšlenkou PDF/A bylo vytvořit podmnožinu formátu PDF takovou, aby dokumenty uchovávané v tomto formátu byly zobrazitelné i po velmi dlouhé době.

Na první pohled by se mohlo zdát, že se jedná jen o takovou podmnožinu jazyka PDF (Obrázek 12.12aby i po mnoha letech bylo možné PDF soubor věrně zobrazit. Avšak PDF/A klade důraz i na druhý aspekt: soubor formátu PDF/A by měl v sobě obsahovat všechny informace, které budou v budoucnu třeba k zobrazení dokumentu. Tj. aby se po letech nezjistilo, že tam něco chybí, a proto dokument nelze věrně zobrazit. Např. nemůžeme se spoléhat na to, že se k zobrazení dokumentu budou využívat nějaké technické vlastnosti nyní používaného zařízení. To se týká např. barev, které musí být vždy specifikovány. Rovněž musí být zcela popsány fonty a obecně veškerá data.

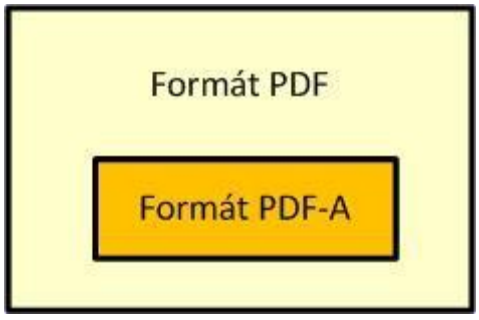
Zakázány jsou rovněž externí odkazy (stejné po mnoha letech archivace už nebudou funkční).

Zatímco syntaxi odpovídající konkrétní podmnožině standardu PDF lze v principu jednoduše ověřit, tak požadavek na úplnost informací se může jevit snadným, ale po čase si můžeme uvědomit, že nám něco připadalo samozřejmé a do dokumentu jsme to opomněli vložit.

Formát PDF/A specifikuje standard ISO 19005. Postupně byly publikovány verze tohoto standardu uvedené v tabulce 1. Pokud se podíváte dovnitř do souboru např. PDF-3/B, pak soubor bude začínat záhlavím:

%PDF-1.7

Tj. jedná se o soubor formátu PDF 1.7 s tím, že při jeho vytvoření byly (doufejme) splněny podmínky standardu ISO 19005-3 pro formát PDF-3/B. To, že se jedná o formát PDF-3/B je pak vyznačeno v metadatech souboru.



Obrázek 12.12 Formát PDF/A specifikuje podmnožinu formátu PDF

Tabulka 12.1 Přehled PDF/A formátů

Verze standardu	**Rok publikace**	**Vychází z formátu**	**Zavádí PDF/A formáty**
ISO 19005-1	2005	PDF 1.4	PDF-1/A PDF-1/B
ISO 19005-2	2011	PDF 1.7 (ISO 32000-1)	PDF-2/A PDF-2/B PDF-2/U
ISO 19005-3	2012	PDF 1.7 (ISO 32000-1)	PDF-3/A PDF-3/B PDF-3/U

V názvech jednotlivých formátů PDF/A jsou za lomítkem písmena A (All), B (Basic) a U (Unicode) vyjadřující úroveň shody se standardem ISO 19005:

- Úroveň B je nejpřísnější a měla by zaručovat dlouhodobou archivaci dokumentů. Tato úroveň neobsahuje logickou strukturu dokumentu. Subjektivně bychom tuto úroveň popsali, že od „scanovaných a nevytěžovaných“ dokumentů se liší zejména tím, že v textu dokumentu je možné vyhledávat a text je možné označit a kopírovat do schránky.
- Úroveň A umožňuje vše, co standard ISO 19005 povoluje, tj. zejména obsahuje logickou strukturu dokumentu. S dokumentem se tak pohodlněji uživatelsky pracuje.
- Úroveň U byla zavedena později. Je obdobná úrovni B s tím, že obsahuje navíc tzv. podporu Unicode. Tím se rozumí, že označený text lze kopírovat v Unicode.

Tabulka 12.2 Podpora vybraných vlastností v jednotlivých verzí PDF/A

Vlastnost	**PDF/A-1**	**PDF/A-2**	**PDF/A-3**
JavaScript	Ne	Ne	Ne
Multiméda	Ne	Ne	Ne

Šifrování	Ne	Ne	Ne
Transparent-nost	Ne	Ano	Ano
Vrstvy	Ne	Ano	Ano
JPEG2000	Ne	Ano	Ano
Vložené soubory formátu PDF/A	Ne	Ano	Ano
Libovolné vložené soubory	Ne	Ne	Ano
Podpis PAdES	Ne	Ano	Ano
Podpis PKCS#7	Ano	Ano	Ano

12.6.1 Je soubor opravdu formátu PDF/A?

Může se zdá, že při vytváření PDF/A dokumentů stačí jen zvolit verzi formátu PDF/A, a je hotovo. Není tomu tak. Musíme ještě verifikovat, jestli náš software vygeneroval soubor opravdu v příslušném PDF/A formátu. Obdobně jako u elektronického podpis nestačí, že dokument je podepsán, ale podpis ještě musíme rovněž verifikovat (ověřit).

12.7 PAdES

Zaručený elektronický podpis pro dokumenty v PDF formátu se označuje jako PAdES (PDF Advanced Electronic Signatur). PAdES je specifikován v ETSI TS 102 778-1 až -6. Pokud někdo očekává, že PAdES bude prostým mechanickou aplikací CMS podpisů, na podpisy s příponami -BES, -PES, -T až –A, tak bude zklamán. PAdES vyhodnotil zkušenosti s formáty zaručených podpisů a vše upravil trochu podle svého. Konkrétně zavádí následující typy PAdES podpisů:

- PAdES–CMS – podpis, který je specifikován v ISO 32000-1;
- PAdES-BES;
- PAdES-EPES;
- PAdES-LTV (Long Term Validation);
- PAdES pro XML obsah.

Z hlediska úrovně podpisů B, T, LT a LTA:

- Formát PAdES-CMS není z hlediska úrovně podporován.
- Úroveň B (basic Level) – odpovídají podpisy PAdES-BES a PAdES-EPES pokud neobsahují časové razítko, ale to je přitom pro tyto formáty doporučené..
- Úroveň T (Trusted time for signature existence) – odpovídají podpisy PAdES-BES a PAdES-EPES s elektronickým podpisem CAdES-T, tj. doplněné o časové razítko z elektronického podpisu.
- Úroveň LT (Long Term level) - elektronický podpis doplněný o materiál (certifikáty, CRL, OCSP odpovědi atp.) nutný pro verifikaci podpisu. Této úrovni odpovídá PAdES-LTV.
- Úroveň LTA (Long Term with Archive timestamps) – odpovídá PAdES-LTV.

12.7.1 PAdES-CMS

PAdES-CMS (nebo též PAdES Basic) je popsán přímo v ISO 32000-1 s tím, že doporučuje, aby obsahoval revokační informace a časové razítko podpisu. Jelikož ale nemůže počítat s tím, že by tyto údaje byly ve struktuře CMS podpisu (ty jsou až v podpisu CAdES), tak se počítá s tím, že tyto údaje budou uvedeny pomocí prostředků PDF.

12.7.2 PAdES-BES a PAdES-EPES

Asi největším překvapením je, že oba tyto formáty nejsou přesnou obdobou formátu CAdES-BES a CAdES-EPES. Formáty PAdES-BES a PAdES-EPES ve své podstatě odpovídají formátu spíše CAdES-T, protože tyto podpisy mohou obsahovat nepodepisovaný atribut Časové razítko podpisu. Tj. důkaz, že dokument byl podepsán v určitém čase. Formáty PAdES-BES a PAdES-EPES jsou tak ty formáty elektronických podpisů v PDF, které jdou pro dnešní praxi nejaktuálnější.

Formáty PAdES-BES a PAdES-EPES již předpokládají, že v PDF dokument neobsahuje jen prostý CMS podpis, ale podpis CAdES. Tj. podpis může obsahovat podepisované a nepodepisované atributy zavedené standardem CAdES .

Formáty PAdES-BES a PAdES-EPES se liší v tom, že PAdES-EPES obsahuje navíc podepisovaný atribut Politka elektronického podpisu (Signature Policy). Oba však:

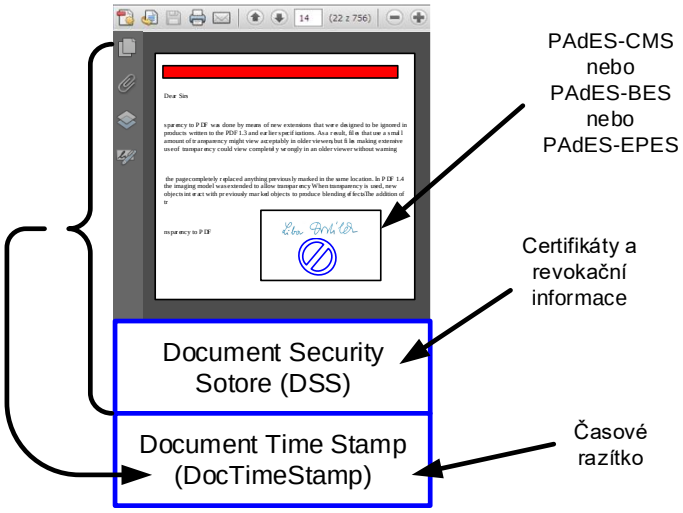
- musí obsahovat podepisovaný atribut Identifikátor certifikátu pro ověřování podpisu (Signing Certificate V2);
- volitelně mohou obsahovat nepodepisovaný atribut Časové razítko podpisu (Signature timestamp).

12.7.3 PAdES-LTV

Tento typ podpisu řeší problém doby platnosti časového razítka, resp. doby platnosti certifikátu autority pro vydávání časových razítek (TSA). Je tedy analogií podpisu typy CAdES-A, má však odlišný formát.

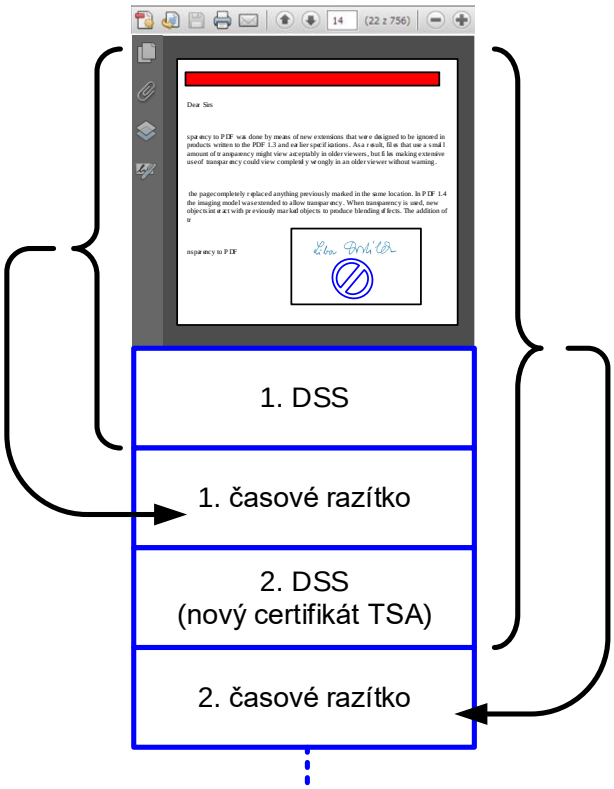
PAdES-LTV jde formou rozšíření samotného PDF formátu. Konkrétně rozšiřuje PDF specifikaci dva objekty, které se vkládají do PDF dokumentu (Obrázek 12.13):

- Objekt Document Security Store (DSS), který obsahuje certifikáty a revokační informace (CRL a OCSP odpovědi) potřebné k verifikaci podpisů.
- Objekt Dokument Time Stamp (DocTimeStamp) obsahující časové razítko ze všech ostatních částí dokumentů, včetně elektronických podpisů.



Obrázek 12.13 Objekty DSS a TS

V případě blížícího se vypršení certifikátu autority pro vydávání časových razítek (TSA) se na konec dokumentu přidá další dvojice DSS a časové razítko ze všech předchozích částí dokumentu (Obrázek 12.14), čímž se docílí dlouhodobá ověřitelnost elektronického podpisu (Long Term Validation).



Obrázek 12.14 Formát PAdES-LTV

13 Čipové karty

Soukromé klíče, sídelná tajemství či jiný kryptografický materiál tvoří aktiva, která je třeba proti případným hrozbám chránit odpovídajícími opatřeními. Nejjednodušší metodou uložení aktiv je jejich uložení na lokální disk. Nevýhodou ale je, že data lokálního disku lze poměrně snadno zcizit. Nebezpečí ale např. spočívá v aplikacích typu „trojský kůň“, které mohou být schopny zjistit přístupové heslo k aktivu nebo přečíst přímo rozšifrovanou podobu aktiva ve chvíli, kdy je používáno v paměti počítače. Takové trojské koně mohou být staženy např. z internetu nebo získány elektronickou poštou. Jiným způsobem útoku je modifikace aktiva na lokálním disku a řada dalších.

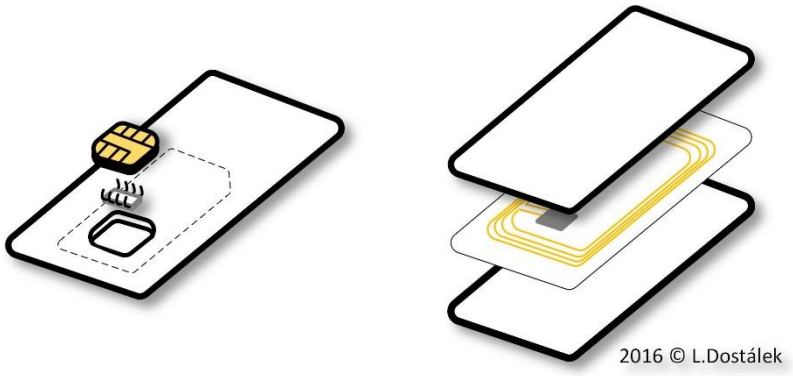
Obdobným problémem je předání počítače (resp. mobilní zařízení) do opravy či přístup administrátorů na tato zařízení. Tady si může být uživatel jist, pouze pokud svá aktiva uloží mimo lokální (neodnímatelné) úložiště – např. na čipovou kartu – a při vydání zařízení z vlastních rukou, fyzicky vyjme tato aktiva ze zařízení.

Nejběžnějším typem ochrany osobních aktiv je jejich uložení do specializovaných hardwarových zařízení, někdy označovaných jako hardwarové klíče (čipové karty, autentizační kalkulátory, HSM apod.)

Autentizačními kalkulátory rozumíme samostatné technické zařízení přímo nepropojené s počítačem, které slouží pro generování jednorázových hesel pro autentizaci držitele kalkulátoru nebo autentizaci dat zasílaných držitelem kalkulátoru. Autentizační kalkulátory jsou tak elektronické pomůcky pro autentizaci klienta (případně pro autentizaci dat zadaných klientem).

Zatímco autentizační kalkulátory a čipové karty zpravidla slouží pro autentizaci konkrétního uživatele (člověka), tak tak HSM (*Host Security Module*, někdy též *Hardware Security Module*) slouží zejména pro ochranu aktiv serverů, které často požadují paralelně vykonat velké množství kryptografických operací. HSM moduly jsou specializované zařízení (počítače), které zajišťují také oddělení rolí administrátorů těchto zařízení od rolí pracujících s kryptografickým materiálem, tj. tzv. *segregation of duty*.

Čipová karta *(Integrated Circuit Card - ICC*) je plastiková karta, která má ve svém těle vložen čip. Nejčastější technologii vložení čipu do karty je vyfrézování dutiny v těle karty a následné vlepení čipu s kontakty do této dutiny. V případě bezkontaktních čipových karet se se karta skládá ze dvou těl, mezi která se vloží velice tenký *inlay*, který obsahuje čip a po obvodu několik závitů tvořících anténu (obr. 13.1).



obr. 13.1 Kontaktní a bezkontaktní čipová karta

Čipová karta může obsahovat jak kontaktní část, tak i bezkontaktní část. Takové karty nazýváme buď hybridními, nebo duálními:

- Hybridní čipové karty obsahují dva na sobě nezávislé čipy (kontaktní a bezkontaktní).
- Duální čipové karty obsahují jeden čip, který má dva výstupy (kontaktní a bezkontaktní).

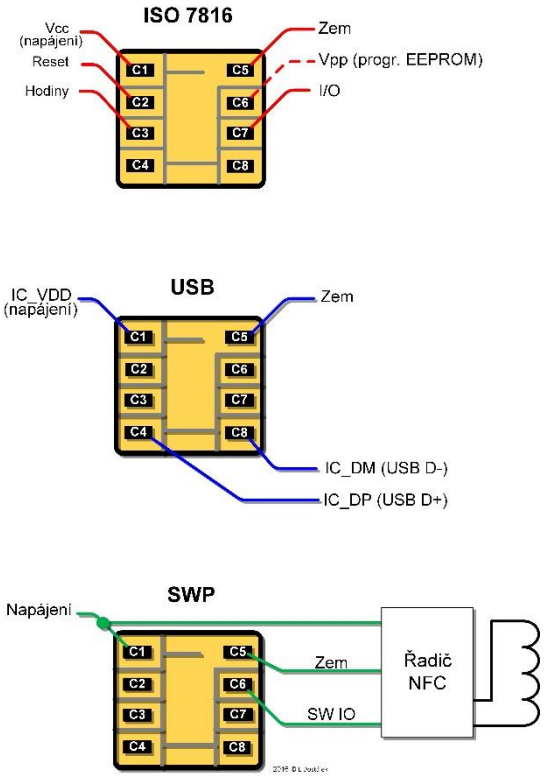
Pro komunikaci s terminálem (čtečkou) nepotřebují bezkontaktní čipové karty galvanický spoj, komunikují pomocí elektromagnetických vln. Napájení rovněž obstará čtečka (terminál) na bázi přenosu energie pomocí indukce. Kontaktní čipové karty mají na sobě kontakty, pomocí kterých se propojují se čtečkou. Napájení rovněž obdrží z terminálu (čtečky) pomocí kontaktů C1 a C5 (obr. 13.2).

V současné době se nejčastěji využívají následující standardy:

- ISO 14443 – standard pro bezkontaktní karty na frekvenci 13,56 MHz pracující přibližně do

vzdálenosti 10 cm. Tento standard má historický základ ve firemním standardu firmy Philips Electronics pod označením MIFARE™. Hovorově se proto někdy o těchto čipových kartách stále někdy mluví jako o kartách MIFARE™.

- ISO 15693 – standard pro bezkontaktní karty na frekvenci 13,56 MHz pracující až do vzdálenosti 1-1,5 m.
- ISO 7816 – standard pro kontaktní karty. Nestandardizuje jen kontakty a jejich elektronické vlastnosti, ale i fyzické vlastnosti karet, datové příkazy, aplikační elementy atd. V těchto oblastech se často používá i pro bezkontaktní karty. Tento standard se skládá z následujících částí:
  - ISO 7816-1 specifikuje fyzikální charakteristiky karty (tepelnou odolnost, ohebnost karty, odolnost proti rentgenovému záření, UV záření, elektromagnetickému poli, minimální počet zasunutí karty do čtečky apod.).
  - ISO 7816-2 specifikuje umístění kontaktů na kartě, jejich rozměr a funkci.
  - ISO 7816-3 specifikuje elektrické signály a přenosové protokoly. Specifikuje dále zmíněné protokoly T=0, T=1, až T=15.
  - ISO 7816-4 specifikuje datové příkazy pro komunikaci s kartou, přístupové metody k datům na kartě, zabezpečení komunikace (*secure messaging*) mezi čtečkou a kartou atd.
  - ISO 7816-5 Registrace aplikací
  - ISO 7816-6 Aplikační datové elementy
  - ISO 7816-7 Příkazy jazyka *Structured Card Query Language* (SCQL)
  - ISO 7816-8 Příkazy pro bezpečnostní operace
  - ISO 7816-9 Příkazy pro správu karty
  - ISO 7816-10 Elektronická signalizace pro synchronní karty
  - ISO 7816-11 Osobní identifikace pomocí biometrických metod
  - ISO 7816-12 Komunikace s kontaktní kartou s využitím USB
  - ISO 7816-13: Příkazy aplikačního managementu muliti-aplikačního prostředí
  - ISO 7816-15 Standardní aplikace pro uložení kryptografických informací
- ETSI *(European Telecommunications Standards Institute)* vydala celou řadu standardu pro čipové karty používané zejména v mobilních zařízeních označované jako UICC. Zkratkou UICC



obr. 13.2 Kontakty čipové karty dle ISO 7816-2, USB a SWP (pokud vaše „SIMka“ má jen 6 kontaktů, tak snadno pochopíte, které chybí)

(*Universal Integrated Circuit Card*) se označují čipové katy používané v mobilních zařízeních, kde jsou využívané pro autentizaci (přihlášení) účastníka do sítě. Objevily se s 2G sítěmi. Umožnily oddělit mobilní zařízení od jeho autentizace. Účastník tak může vyjmout kartu ze zařízení a vložit ji do jiného zařízení a autentizovat se do sítě z nového zařízení aniž byla nutná asistence poskytovatele sítě. Tj. účastník si může svá osobní aktiva uložená na UICC ze zatížení vyjmout, pokud zařízení vydává ze své moci.

- GlobalPlatform je organizace, kterou založily společnosti zabývající se platebními kartami a mobilními komunikacemi, tj. business, který asi nejvíce využívá čipové karty. Specifikace GlobalPlatform se zabývají zejména správou obsahu karty, komunikací a bezpečností na úrovni aplikací. Specifikace pojmů jako TEE *(Trusted Execution Environment)* nebo SE *(Secure Element)* najdeme právě v jejích dokumentech.
- Mezinárodní organizace pro civilní letectví *(International Civil Aviation Organization - ICAO)* je mezivládní organizace přidružená k OSN,

která se kromě civilního letectví zabývá i standardizací cestovních dokladů. Konkrétně její dokument 9303 specifikuje mj. elektronickou část cestovních dokladů. Cestovní doklady nemají tvar čipové karty, ale jejich inlay je „zatažen“ buď v datové stránce, nebo v deskách cestovního pasu. Dokument 9303 využívá standard ISO 14443 pro radiovou komunikaci, standard ISO 7816 pro vyšší vrstvy, ale dokument 9303 už nemá žádnou souvislost např. se standardy GlobalPlatform. Dokument 9303 popisuje, jak budou v čipu uloženy základní identifikační údaje držitele a také určuje, jak budou uloženy jeho biometrické údaje a případně další údaje.

- Evropský výbor pro normalizaci (*Comité Européen de Normalisation – CEN*) vydal řadu standardů CEN/TS 15480, které specifikují tzv. Evropskou občanskou kartu (občanský průkaz). Standardy CEN/TS 15480 se odkazují i na dokument 9303. Uvedené normy CEN jsou přeložené do češtiny jako ČSN, ale jsou placené.

Standardy ICAO a CEN jsem uvedl proto, že čas od času někdo přijde s nápadem implementovat cestovní doklady nebo občanský průkaz v „mobilním telefonu“. Je vcelku zjevné, že technicky tomu v podstatě nic nebrání, ale problém je právní, protože mezi vlastníkem mobilního zařízení a držitelem cestovních dokumentů (resp. občanského průkazu) může být velký rozdíl, byť obojí se může nacházet ve stejné kapse kalhot. To už vůbec neuvažuji, že by byl občanský průkaz implementován v eSE *(Embedded Secure Element)* a já potřeboval dát mobil do opravy!

1.1 Kontakty čipové karty

Kontaktní čipové karty mají dle ISO 7816-2 osm kontaktů. Tento standard předepisuje osm plošek C1 až C8 o rozměru 2x1,7 mm, kde musí být umístěny kontakty (viz obr. obr. 13.2). Výrobci pak vytváří kontakty o trochu větším rozměru (zlaté plochy na obr. 13.2) tak, aby tvořily módní design karty.

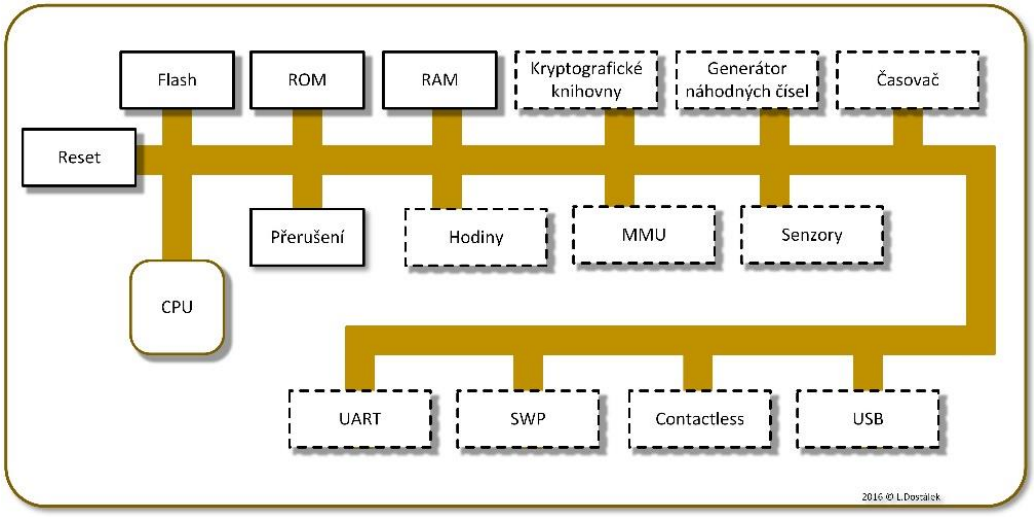
Podle napájecího napětí (V_{CC}) dělí ETSI TS 102 221 karty do následujících tříd:

- Třída A (V_{CC} mezi 4,5 a 5,5 V). Třída A nepředpokládá využití kontaktů C4 a C8.
- Třída B (V_{CC} mezi 2,7 a 3,3 V), pokud jsou využity kontakty C4 a C8 pro USB interface, pak se pro tyto kontakty použije napěťová třída Inter-Chip USB 3,0 V.
- Třída C (V_{CC} mezi 1,62 a 1,98 V) , pokud jsou využity kontakty C4 a C8 pro USB interface, pak se pro tyto kontakty použije napěťová třída Inter-Chip USB 1,8.

Na obr. 13.2 je rovněž znázorněno zapojení kontaktů v případě rozhraní USB a SWP. Rozhraní IC_USB a SWP budou zmíněna později.

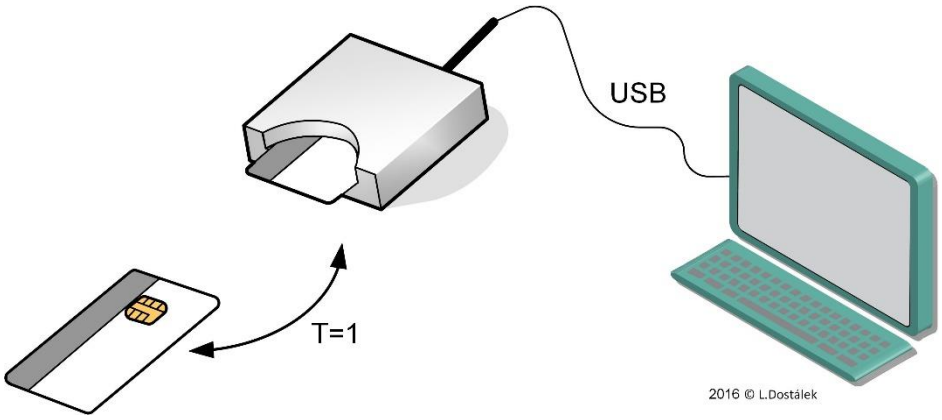
1.2 Logické schéma čipové karty

Nejjednodušší čipové karty byly osazeny pouze paměťovými registry, které je možné nastavovat, přičítat k nim, odečítat od nich apod. Na rozdíl od nich procesorové karty mají kromě paměti i jednočipový procesor schopný vykonávat příkazy. Procesor je řízen operačním systémem karty.



obr. 13.3 Blokové schéma čipové karty

Procesorové čipové karty se mohou skládat z řady modulů (obr. 13.3). Vedle CPU, pak obsahují zejména paměť. V paměti ROM je zpravidla uložen



obr. 13.4 Mezi kartou a čtečkou je jeden komunikační protokol (např. T=1) a mezi čtečkou a počítačem je druhý komunikační protokol (např. USB)

operační systém, který se zavádí do RAM. Data se ukládají do souborových systémů realizovaných v paměti Flash (v minulosti se využívala i paměť EEPROM). V případě, že se čipová karta používá pro autentizaci či autorizaci, pak jsou též důležité kryptografické knihovny. Nejčastější kryptografické knihovny jsou pro algoritmy: DES/3DES, RSA a Eliptické křivky. Čipové karty pak obsahují jednu nebo více těchto knihoven.

Jak již bylo zmíněno, čipová karta může mít jeden nebo více vstupů/výstupů:

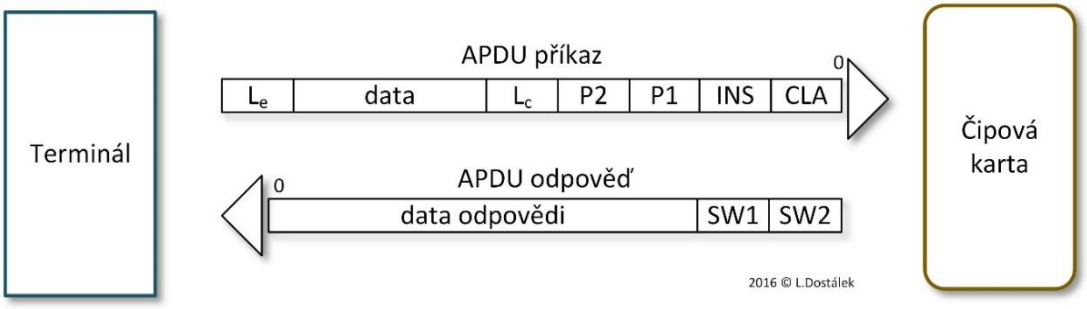
- Sériový vstup/výstup (I/O – ploška C7), k tomu slouží modul UART (*Universal Asynchronous Receiver Transmitter*). Je to jednodrátová obdoba, dnes již nepoužívaných COM portů osobních počítačů.
- USB (ETSI TS 102 600)
- Bezkontaktní (ISO 14443)
- SWP (ETSI TS 102 613) pro propojení s řadičem NFC.

Z hlediska spouštění aplikací v čipových kartách můžeme též karty dělit na statické s pevně nahranými aplikacemi (většina firemních operačních systémů jednotlivých výrobců karet) a dynamické, do kterých je možné vkládat nejen data, ale i spustitelný kód. Nejznámějšími technologiemi dynamických karet jsou systémy JavaCard™ či Multos™.

Pokud se v paměti karty spouští více aplikací, pak modul MMU (*Memory Management Unit*) monitoruje dodržování hranic jednotlivých aplikací v paměti. Před spuštěním každé aplikace jsou totiž stanoveny hranice oblasti paměti, kterou může aplikace využívat. Tuto hranici nelze za běhu aplikace změnit, tj. každá aplikace je zapouzdřená do své oblasti. Bariery na hranicích těchto oblastí se nazývají firewally.

1.3 Terminály

Čtečka čipových karet se správně označuje jako terminál. Jedná se o zařízení, které zprostředkovává komunikaci s čipovou kartou. Terminál může být jako samostatné zařízení, nebo může být propojen např. s počítačem. Pro propojení terminálu s počítačem se často využívá jiný komunikační protokol než ten, který využívá terminál pro komunikaci s kartou. Např. na obr. 13.4 komunikace mezi kartou a čtečkou probíhá protokolem T=1, kdežto čtečka s počítačem komunikuje protokolem USB.



obr. 13.5 Komunikace APDU příkazy

V případě vestavěného terminálu do zařízení se pak použije komunikace po interní sběrnici, např. I^2C, HCI, SPI apod.

Pro zajištění konverze komunikačních protokolů jsou čtečky často realizovány jednočipovým procesorem, což patřičně navyšuje cenu čteček. Nejčastějšími komunikačními protokoly mezi čtečkou a kartou na fyzické vrstvě jsou:

- **T=0** – jedná se o znakově orientovanou poloduplexní sériovou výměnu dat mezi terminálem a kartou.
- **T=1** - jedná se rovněž o sériový protokol mezi terminálem a kartou, ale tentokrát blokově orientovaný, tj. mezi terminálem a kartou se přenáší data po celých blocích dat. Tento protokol zrychluje výslednou komunikaci s kartou, avšak karta musí disponovat větší RAM pro vyrovnávací paměti. Dnešní karty umí současně jak protokol T=0, tak i protokol T=1.
- **Bezkontaktní přenos,** někdy hovorově označovaný jako T=CL *(Contact Less)*.
- **USB,** někdy hovorově označovaný jako T=USB. Viz odstavec 13.1.

Velice důležitým parametrem čteček se stává signalizace vytažení karty z terminálu, který se ocení zejména v případě přihlašování k počítači (do Windows, k Linuxu apod.) pomocí čipové karty. V případě vysunutí karty z terminálu automaticky dojde k zablokování stanice. Bohužel mobilní zařízení tento test často nepoužívají.

V případě mobilních telefonů se zpravidla mobilní telefon chápe jako terminál. To pochopitelně neplatí pro NFC.

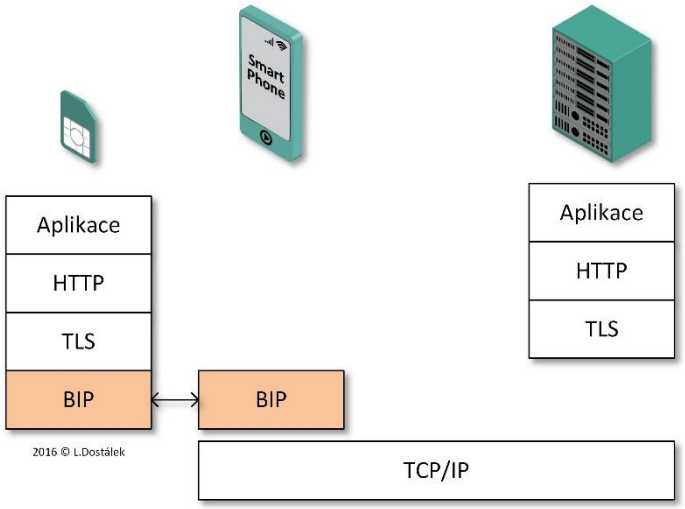
1.4 ATR

Čipová karta ISO 7816 po svém vložení do terminálu vrací tzv. ATR řetězec *(Answer To Reset)*. ATR je 2 až 33 B dlouhý řetězec, který nastaví již výrobce karty. Na základě ATR je často možné odlišit různé druhy karet, není to však pravidlem. Jestliže má operační systém pracovat s konkrétním typem karty, zpravidla to znamená, že pracuje s kartou o tom a tom ATR řetězci. U ISO 7816 karet se používá jako základní test jejich funkčnosti skutečnost, jestli po přivedení napájecího napětí vrací ATR.

ATR má interní strukturu, ze které lze vyčíst řadu vlastností karty: podporované komunikační (T=0, T=1 atd.), maximální doba čekání v protokolu T=0, maximální velikost bloku pro protokol T=1, maximální frekvenci hodin atd.

U bezkontaktních karet se používá termín ATS *(Answer To Select)*. Význam je obdobný jako ATR.

1.5 APDU



obr. 13.6 Síťový model protokolu BIP

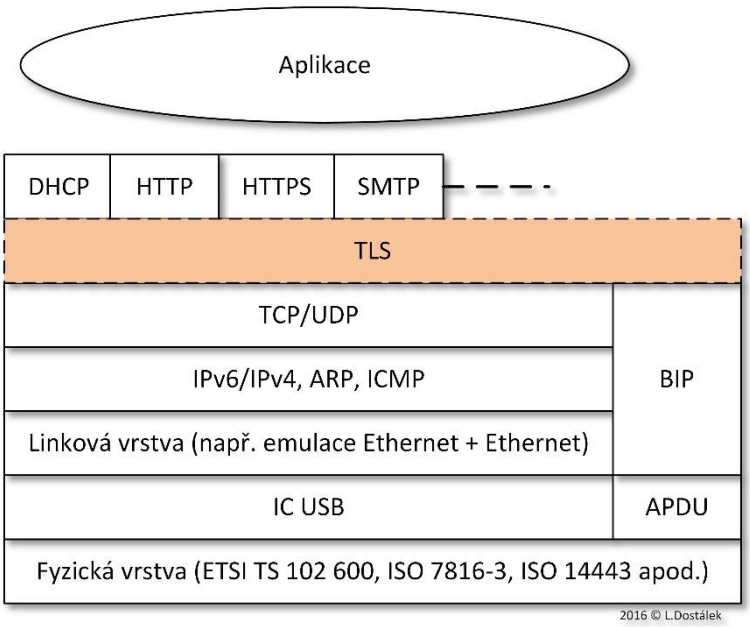
Protokoly na fyzické vrstvě řeší pouze fyzickou komunikaci mezi kartou a terminálem. Nad těmito protokoly se mezi aplikací v počítači/mobilu a kartou přenáší datové pakety nazývané APDU (*Application Protocol Data Unit*). Pomocí APDU se zasílají instrukce kartě, která vrací odpovědi. APDU je nízko úrovňové rozhraní, pomocí kterého již s kartou mohou komunikovat specializované aplikace v počítači/mobilu, může probíhat personalizace karet apod.

Komunikace APDU příkazy je znázorněna na obr. 13.5, význam jednotlivých polí:

- CLS - třída instrukce (0x80 - proprietární, 0x00 - standardní)
- INS - instrukce
- P1 - parametr 1
- P2 - parametr 2
- Lc - velikost dat (bajty) předávaných do karty v rámci příkazu
- Le - velikost dat očekávaných jako odpověď
- DATA IN - data vstupující do karty
- DATA OUT - data, která jsou výsledkem vykonání příkazu
- SW1,SW2 - návratový kód příkazu (OK nebo indikace chyby)

Příklad příkazu APDU (šestnáctkově): 00 A4 00 00 02 3F 00

- 00 – třída instrukce (standardní instrukce)
- A4 – instrukce SELECT
- 00 00 – parametry P1 a P2
- 02 – do karty budou předávaný 2B
- 3F 00 – předávána data, konkrétně se jedná o identifikaci souboru, která je 3F00, tj. MF (*Master File*) - Jednalo se tedy o APDU příkaz SELECT MF.



obr. 13.7 Síťový model čipové karty (TLS vrstva je volitelná)

Takto jednoduchá komunikace se používá v případě, že kartu využívá jen jedna aplikace. Dnešní karty podporují komunikaci s více aplikacemi. Pro každou takovou komunikaci se vytvoří mezi aplikací a kartou tzv. logický kanál. V APDU příkazu se pak musí vyjádřit číslo kanálu ke kterému APDU příkaz patří. K tomu se využívá pole CLS (třída instrukce) do které se kóduje číslo logického kanálu. Vždy máme k dispozici základní *(basic)* logický kanál číslo 0. Další logické kanály jsou podle ISO/IEC 7816-4 číslovány od 1 až teoreticky do 19. Tyto logické kanály se otevírají APDU příkazem MANAGE CHANNEL.

Velice důležité je, že pokud se zabezpečuje komunikace mezi kartou a terminálem, pak se zpravidla zabezpečuje na úrovni jednotlivých kanálů APDU příkazem MANAGE SECURE CHANNEL *(Application to Application Security)*. Je možný ale i případ, že se zabezpečuje celková komunikace mezi kartou a terminálem *(Platform to Platform)* – např. pokud karta neumožňuje zřizovat logické kanály, tj. podporuje pouze základní logický kanál 0.

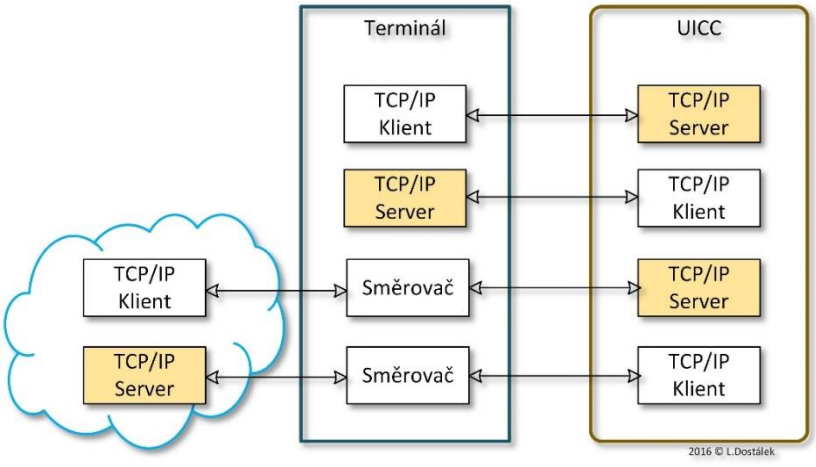
13.1 USB

USB komunikaci mezi kartou a terminálem specifikuje ETSI TS 102 600 (kontakty viz obr. 13.2). Tento standard využívá USB 2.0 s dodatkem *Inter-Chip USB supplement to the USB 2.0 Specification* (dnes existuje i *Inter-Chip Supplement to the USB Revision 3.0 Specification*). Zkráceně se tento standard označuje jako IC USB (dále budu v této publikaci uvádět jen USB).

Při pohledu na obr. 13.2 snadno zjistíme, že kontakty pro USB nekolidují s kontakty ISO 7816. Existují terminály, které podporují USB i ISO 7816. Terminály podporující USB by měly podporovat i ISO 7816 komunikaci. Při zasunutí karty do terminálu pak dojde k volbě komunikačního protokolu..

USB komunikace mezi terminálem a kartou může být využita dvěma způsoby:

- Čipová karta se chová jako USB zařízení. V tomto případě není třeba používat APDU.
- USB se jen pro přenos APDU příkazů (tzv. APDU přes USB). Tato možnost je vhodná zejména pro již existující aplikace, které jsou postaveny na APDU.



Obrázek 13.8 Webový server a čipová karta

1.6 SWP

SWP *(Single Wire Protocol)* specifikuje standard ETSI 102 613. Zapojení kontaktů je zobrazeno na obr. 13.2. Při pohledu na tento obrázek snadno zjistíme, že kontakty nekolidující s IC USB kontakty.

SWP je odvozen od dnes již historického protokolu HDLC *(High-Level Data Link Control)*. SWP se využívá pro komunikaci mezi kartou *(secure element)* a řadičem NFC. Tyto dva hardwarové prvky jsou třeba např. pro aplikace emulující bezkontaktní platební karty pomocí mobilního zařízení.

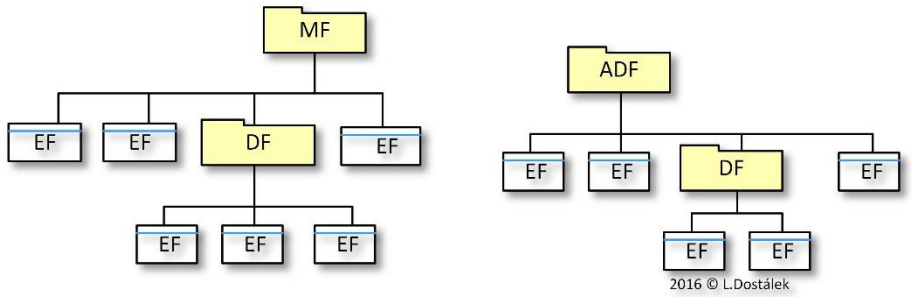
Je třeba ještě dodat, že secure element pro NFC může být implementován i v mikro SD kartě nebo i vestavěn přímo na základní desce mobilního zařízení. V případě mikro SD karty se rovněž používá protokol SWP.

1.8 BIP

BIP *(Bearer Independent Protocol)* je historickým mezičlánkem pro karty, které nemají implementované TCP/IP, ale mají implementovány nějaké aplikační funkce (např. webový server). TCP/UDP komunikace je pak zakončena na terminálu a mezi ním a kartou se pro přenos dat provádí pomocí APDU příkazů OPEN CHANNEL, CLOSE CHANNEL, SEND DATA, RECEIVE DATA a GET CHANNEL STATUS (obr. 13.6). Dalo by se říci, že TCP/IP rozhraní je z karty exportováno do terminálu.

1.9 Webový server

Máme-li na kartě TCP/IP, můžeme implementovat i webový server. Jak je znázorněno obr. 13.8, tak z hlediska architektury klient/server jsou na čipové kartě přípustné všechny eventuality. Jako server si nyní představíme webový server.



obr. 13.9 Příklad struktury dat na čipové kartě

1.7 Emulace Ethernetu

Máme-li připojenou kartu s terminálem pomocí USB, pak emulace protokolu Ethernet přes USB je alternativou k používání APDU příkazů.

Protokol emulace protokolu Ethenrnet přes USB je specifikován v dokumentu *Universal Serial Bus Communications Class Subclass Specification for Ethernet Emulation Model Devices*. Protokol emulace protokolu Ethenrnet přes USB je třeba k vyřešení problému, kterým je, že rámce USB jsou kratší než rámce protokolu Ethernet. Nezbývá než ethernetové rámce nakrouhat na menší části, které se doplní krátkým záhlavím emulačního protokolu a vloží do USB rámce. Záhlaví emulačního protokolu specifikuje, kterému rámci protokolu Etherent patří který USB rámec.

Máme-li mezi terminálem a kartou komunikaci protokolem Ethernet, pak dalším krokem je vložení IPv4 nebo IPv6 datagramů do těchto rámců. Terminál pak dokonce může pracovat jako směrovač *(router)*. Následně může být implementován síťový model TCP/IP (obr. 13.7).

Pro někoho může být překvapivé, že na čipové kartě muže být implementován webový server, který se často označuje zkratkou SCWS *(Smart Card Web Server)*. Zajímavé je, že se může jednat i o HTTPS server, tj. server využívající zabezpečení TLS protokolem. Pro autentizaci s předem sdílenými klíči – tzv. TLS PSK, tj. *"Pre-Shared Key Ciphersuites for Transport Layer Security"*.

TLS relace může být zřízena nikoliv jen mezi terminálem a kartou, ale i mezi vzdálenou aplikací a kartou. Může být tak vytvořen bezpečný TLS kanál mezi kartou a vzdálenou aplikací. Toho lze využít např. pro vzdálený management karty.

1.10 Zabezpečení komunikace s čipovou kartou

ETSI TS 102 484 rozlišuje 4 způsoby zabezpečení komunikace s čipovou kartou. Důležité rovněž je, mezi kterými entitami se zabezpečení realizuje:

- TLS zabezpečení mezi aplikací v čipové kartě a aplikací v terminálu nebo zařízení připojeném lokálně nebo přes IP protokol. Pro toto zabezpečení je nutné, aby čipová karta přímo podporovala IP protokol (např. přes USB) nebo aby alespoň podporovala BIP.
- IPsec mezi terminálem a čipovou kartou. Tato eventualita je možná jen v případě komunikace pomocí USB.
- Zabezpečení protokolu APDU *(Secured APDU)*. Tato možnost připadá do úvahy v případě ISO 7816 karet nebo karet podporujících IC USB v případě, že se jedná o komunikaci APDU přes USB (nikoliv IP). Zde přichází do úvahy dva případy:
    - Zabezpečení komunikace na úrovni logického kanálu, tj. zabezpečení komunikace mezi aplikacemi v terminálu a v čipové kartě *(Application to Application)*.
    - Zabezpečení komunikace mezi kartou a terminálem *(Platform to Platform)*. Podpora tohoto typu zabezpečení je zpravidla signalizována v ATR.

  Při vytváření bezpečného kanálu pro zabezpečení protokolu APDU *(Secured APDU)* se nejprve vytvoří zabezpečený kanál, a pak se přenáší data. Při vytváření zabezpečeného kanálu zpravidla dochází i k autentizaci. Protokolů pro zabezpečení kanálu existuje celá řada. Od protokolů na bázi symetrické kryptografie, až pro protokoly na bázi asymetrické kryptografie.

1.11 Struktura dat uložených v kartě

Data uložená na čipové kartě jsou organizována do souborových systémů (obr. 13.9). Na kartě je vždy jeden kořenový adresář nazývaný MF (*Master File*), ve kterém mohou být podadresáře – DF (*Dedicated File*) nebo datové soubory EF (*Elementary File*). Původně už podadresáře nemohly obsahovat další podadresáře. Toto omezení padlo, avšak reálně nebývá vnoření hlubší než 3-4.

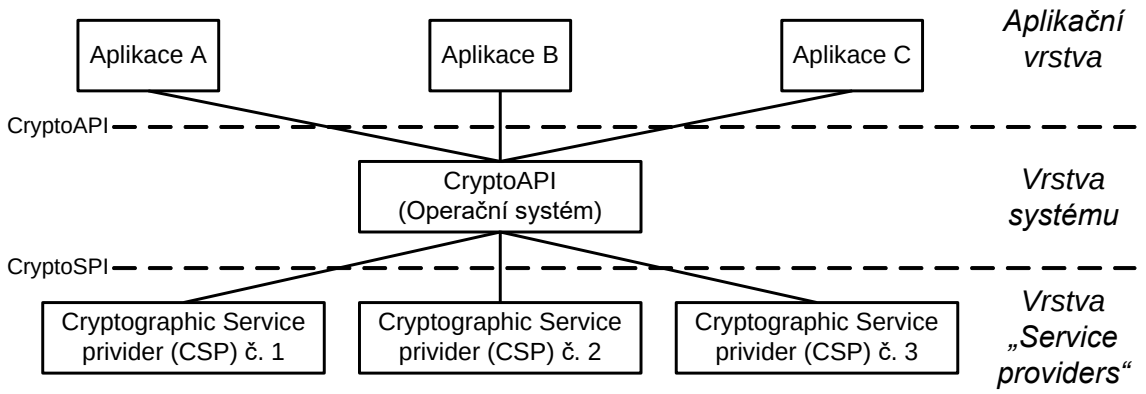
Kromě kořenového adresáře MF může být na kartě jeden nebo více ADF *(Application Dedicated File)*. Je to samostatná souborová struktura, která není zavěšena pod MF. ADF je zpravidla určen k uchovávání dat konkrétní aplikace.

EF se dělí na pracovní a interní. Pracovní EF jsou dostupné např. pomocí APDU příkazů i mimo čipovou kartu. Interní EF naopak takto dostupné nejsou (nejsou „viditelné“ mimo kartu). Do interních EF se ukládají např. soukromé klíče, tajné klíče, sdílená tajemství apod. Klasickým příkladem je čipová karta, která pomocí vlastní kryptografické knihovny sama generuje dvojici veřejný a soukromý klíč pro elektronický podpis. Soukromý klíč uloží do interního souboru a veřejný do pracovního souboru. Soukromý klíč pak nikdy nepouští kartu.

13.2 Čipová karta a operační systém (middleware)

Základním problémem je skutečnost, že čipové karty včetně middleware dodává často jiný výrobce než výrobce operačního systému. Operační systém tedy musí umožňovat do sebe začlenit softwarové moduly třetích stran.

Na obr (Obrázek 13.11) je řešení tohoto problému firmou Microsoft. Aplikace požadující kryptografické funkce je v celku logicky požadují od operačního systému. K tomuto účelu operační systém poskytuje rozhraní CrptoAPI. Takovými funkcemi může být generování párových dat, šifrování, podepisování apod. Jenže ve výsledku tyto operace mohou být zajištěny jak prostředky dodávanými s operačním systémem, tak i prostředky třetích stran. Proto CryptoAPI vysokoúrovňové kryptografické požadavky aplikací dekomponuje na jednotlivé krypto-



Obrázek 13.10 Microsoft řeší podporu čipových karet pomocí CSP

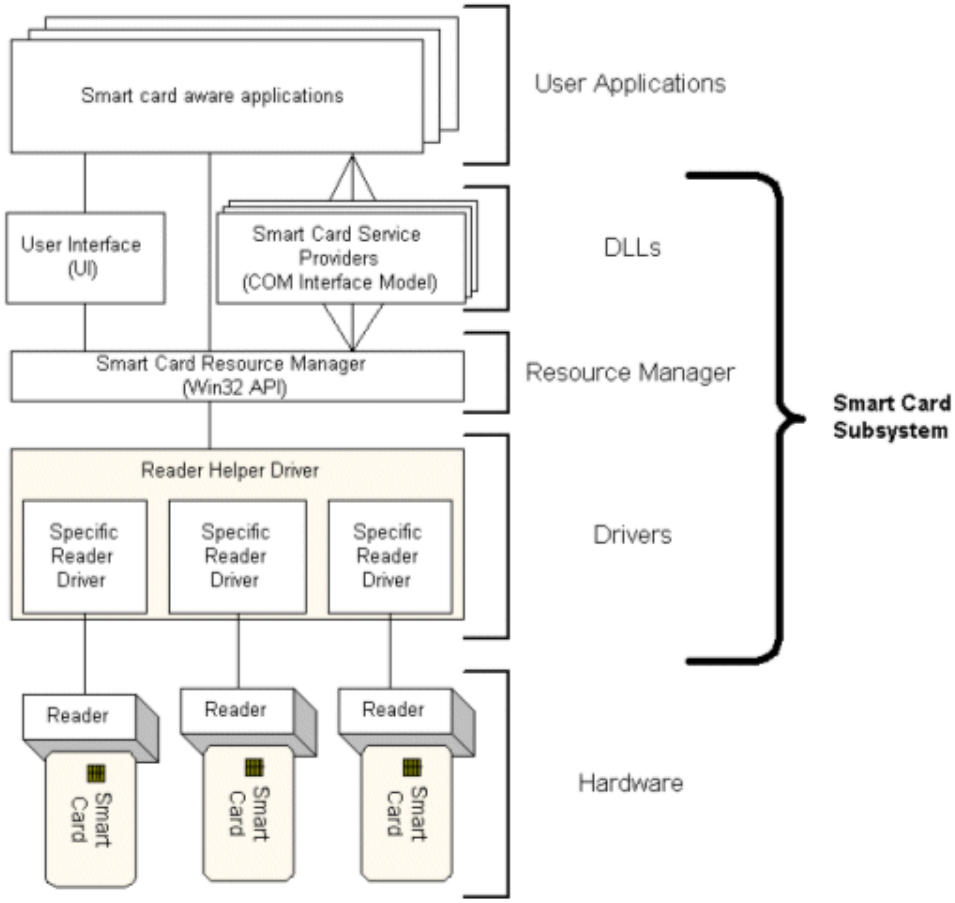
grafické operace, o jejichž provedení požádá konkrétní specializovaný modul – CSP (*Cryptografic Service Provider*). CSP je pak onen kýžený modul, který může být dodán i třetí stranou. Má-li se např. vytvořit elektronický podpis, pak je od CryptoAPI vyžadováno dodání CMS zprávy SignedData. Do této CMS zprávy je nutné vložit soukromým klíčem šifrovaný otisk. Do CSP tak propadne požadavek „soukromým klíčem" šifruj zaslaných 20 B dat otisku. Pokud je CSP realizováno pouze softwarově, nalezne se soukromý klíč na disku a spočte výsledek. Pokud se jedná o CSP pro čipovou kartu, transformuje se tento požadavek na APDU příkaz pro čipovou kartu a skrze čtečku se pošle do čipové karty.

Má-li být kryptografická operace provedena pomocí konkrétní čipové karty, musí být s touto čipovou kartou dodán příslušný CSP*) (DLL knihovna), který musí být předem nainstalován. Microsoft umožňuje do svých operačních systémů začleňovat jen CSP, která jsou elektronně podepsána firmou Microsoft. Pokud chceme využívat čipovou kartu také pro přihlašování do systému, musíme současně s instalací CSP v operačním systému registrovat též ATR, aby systém naše karty znal. CSP je většinou dodáváno ve formě instalačního programu, který provede všechna potřebná nastavení.

Microsoft s operačním systémem dodává sadu svých vlastních softwarových CSP (*Microsoft Basic CSP* s omezenými kryptografickými klíči, *Enhanced Microsoft CSP* s plnými klíči atd.), která nevyužívají čipové karty, ale veškerý kryptografický materiál je uložen na disku. Otázkou do pranice je slovo disk. Soukromé klíče a certifikáty bývají uloženy na lokálním disku. Avšak v případě, že využíváte ActiveDirectory a cestovní profily (*Roaming Profile*), pak i soukromý klíč se může stát součástí cestovního profilu a bude distribuován po celé síti v závislosti na tom, ze kterého počítače se budete přihlašovat do sítě. Pokud tedy chcete mít soukromý klíč i v síti Windows pod vlastní kontrolou, pak se jeho umístění na čipovou kartu jeví dobrým řešením.

Dále Microsoft přibaluje do své distribuce CSP některých výrobců čipových karet, které jsou však většinou nepoužitelné, protože i kdybyste náhodou použili čipovou kartu, pro kterou je CSP dodáván jako součást systému, stejně ji pravděpodobně nepoužijete se souborovou strukturou přímo od výrobce, ale souborovou strukturou navrženou lokálním dodavatelem, tudíž potřebujete i pro tuto kartu CSP od lokálního dodavatele.

Problém je ještě s tím, že CSP musí komunikovat s kartou skrze čtečku, takže situace je ještě komplikovanější (Obrázek 13.11). Každá čtečka musí mít v operačním systému příslušný ovladač. V tom není problém. Problém je v tom, že k počítači mohu mít připojeno více čteček a kartu mohu vsunout do libovolné z nich, proto je mezi ovladače čteček a CSP



Obrázek 13.11 Celková architektura podpory karet v systémech Microsoft (převzato z http://www.microsoft.com/technet)

*) CSP = *Crypto Service Provider*

vložen ještě manažer čteček (*Smart Card Resource Manager*). Kromě CSP skrze manažer čteček budou ke kartám přistupovat i ostatní programy jako např. uživatelské rozhraní či utilita pro práci s kartami. Tato architektura je od verze 2 rozpracována jako průmyslový standard PC/SC (původně se jednalo standard Microsoftu).

Když aplikace pomocí standardu PC/SC hledá konkrétní čipovou kartu (konkrétní ATR), manažer čteček jí pro každou čtečku připojenou k systému sdělí:

- Jestli je čtečku možné využívat v této aplikaci
- Jestli je ve čtečce vložena nějaká karta, když ano, pak jí sdělí ATR této karty.
- Jestli je požadovaný ATR registrován v systému.

Standard PC/SC se využívá nejenom pro PKI čipové karty, ale i pro platební čipové karty EMV a rovněž pro hardwarové klíče ve formě USB tokenu, přičemž všechna tato zařízení spojuje využití komunikace pomocí APDU definovaných v ISO 7816. Standard PC/SCzavádí již zmíněný manažer čteček, který umožňuje snadno implementovat a více aplikacím sdílet čtečky a podobná zařízení. Pro dnešní výrobce čteček je již samozřejmostí dodávat ovladače pro PC/SC architekturu. Výsledkem je, že pomocí PC/SC ovladačů pak snadno začleníme čtečky jednotlivých výrobců.

Standard PC/SC se rozšířil natolik, že z původní mateřské platformy Windows byl portován i do prostředí operačních systémů z rodiny UNIX a to v balíku PCSCLite.

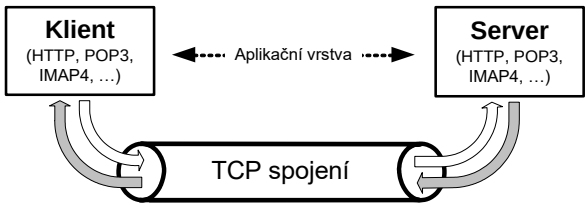
Jediným problémem je, že dosud hojně využívané verze PC/SC nepodporují čtečky s vlastní klávesnicí. Což je problém zejména na registračních autoritách, kdy nechceme, aby držitel karty zadával PIN z klávesnice operátora RA. Problém lze řešit dvěma způsoby:

- Pomoci již zmiňovaného *Secure Messaging*.
- Pomocí čtečky s klávesnicí, která se tváří jako PC/SC avšak v případě, že karta vyžaduje zadání PIN, tento příkaz odchytne, sama vyžádá zadání PIN a výsledek přímo vrátí kartě. Taková čtečka je ale závislá na konkrétních kartách a měla být tedy požadována po dodavateli karet.

Jiným řešením než je CSP je standard PKCS#11. Jedná se o obecně zaměřený standard na zpřístupnění kryptografického hardware a mimo jiné jím lze obsluhovat i čipové karty. Tento standard používají pro obsluhu čipových karet zejména operační systémy UNIX. V operačních systémech Microsoft pak standard PKCS#11 využívají alternativní internetové prohlížeče. Standard PKCS#11 neřeší problematiku manažera čteček, proto i v systémech UNIX se pro správu čteček využívá standard PC/SC.

14 TLS

K zabezpečení přenosu zpráv přes „nebezpečný“ Internet nám může posloužit např. její vložení do CMS obálky (jak je popsáno v následujících kapitolách). V této kapitole se však budeme věnovat zbezpeční toku dat. K tomuto účelu je určen protokol TLS. Předchůdcem protokolu TLS byl protokol SSL (*Secure Sockets Layer*) vytvořený firmou Netscape. Firma Microsoft nezůstala pozadu a vytvořila protokol PCT. V praxi se ujal protokol SSL verze 3. Oficiálním protokolem Internetu se však stal až protokol TLS (RFC-2246: *Transport Layer Security protocol*) vycházející z protokolu SSL verze 3. Aktuální standardem TLS verze 2 je RFC 5246.



Obrázek 14.1 TCP spojení je duplexní

Aplikační protokoly předávají data protokolu TCP, který zajišťují jejich přenos Internetem. Na druhém konci Internetu pak protokoly TCP vracejí přenesená dat aplikacím. I když tvůrci protokolu TLS uvádějí TCP protokol jen jako jeden z možných transportních protokolů, my se zde přidržíme právě protokolu TCP.

Protokol TCP je navržen jako duplexní komunikační protokol, tj. protokol tvořícím dva samostatné kanály. Jeden pro přenos dat z klienta na server a druhý pro přenos ze serveru na klienta (viz obr Obrázek 14.1).

- Protokol TLS tak tvoří vrstvu, která je vložena mezi aplikační protokol a protokol TCP (Obrázek 14.2). Vrstva TLS nikterak nezkoumá data, která jsou jí zasílána aplikační vrstvou – prostě je zabezpečí a předá protokolu TCP. Vrstva TLS tak např. nerozpozná, zdali zabezpečovaná data se skládají z nějakých aplikačních transakcí.
- Vrstva TLS proto ani nemůže provádět zabezpečení na úrovni aplikačních transakcí např. pomocí elektronického podpisu jednotlivých transakcí. Prostě přebírá od aplikační vrstvy paket po paketu a zabezpečuje jednotlivé pakety. U každého paketu je vrstva TLS schopna zajistit jeho privátnost (šifrování) a integritu dat pomocí HMAC (nikoliv pomocí elektronického podpisu). Integrita dat se zajišťuje na základě HMAC počítaného z přenášených dat a sdíleného tajemství. Pokud aplikace vyžaduje elektronický podpis jednotlivých transakcí, musí si jej zajistit sama na aplikační vrstvě.

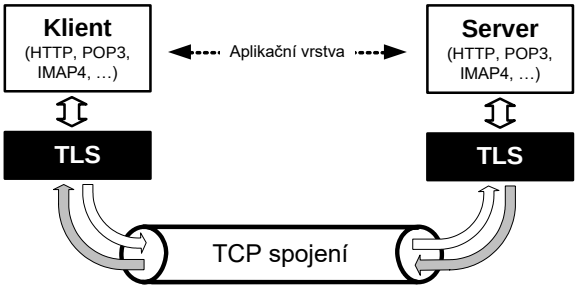
Zvláštností protokolu TLS je formát dat. Ten není ASCII (jedná se přece o šifrovaná data), ale nejedná se ani o BER či DER kódování, protože ta jsou určena zejména pro dávky dat. Data protokolu TLS jsou v podstatě pevného formátu a popisují se jakýmsi intuitivním jazykem vytvořeným jen pro účely protokolů SSL/TLS. My se nebudeme zabývat definicí tohoto popisu a struktury dat si budeme kreslit do obrázků. Bude tak vcelku jasné, co norma míní, pokud bychom však potřebovali danou informaci znát naprosto přesně, pak je nutné sáhnout k RFC-2246.

Další zajímavostí je termín **„elektronický podpis TLS“**, který zde není míněn ve smyslu zákona o elektronickém podpisu, ale ve smyslu algoritmu. Tento elektronický podpis se používá v úvodním dialogu pro autentizaci serveru i klienta.

Protokol je TLS otevřený pro využití libovolným protokolem vyšší vrstvy (libovolným aplikačním protokolem).

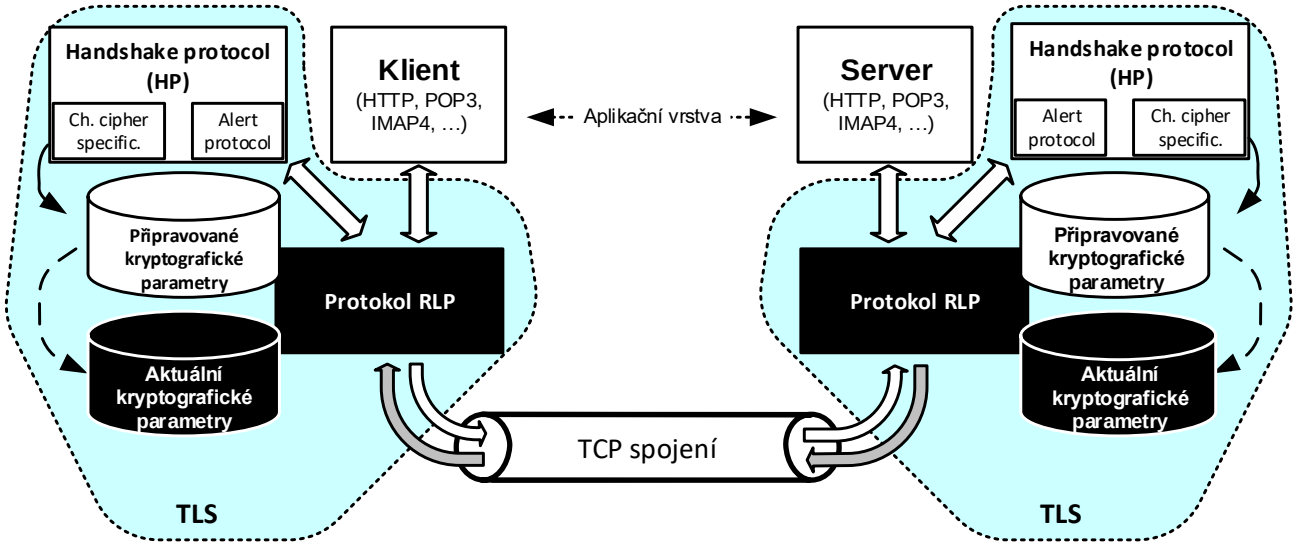
TLS umožňuje:

- Autentizaci serveru i klienta na základě jejich certifikátů.
- TLS sice může komunikovat zcela anonymně, ale zpravidla provádí alespoň autentizaci serveru.
- Server, který se sám autentizuje, ale nevyžaduje autentizaci klienta se nazývá anonymním TLS serverem.



Obrázek 14.2 TLS se vkládá mezi aplikační protokol a protokol TCP

- Šifrování přenášených dat. I když počáteční výměna kryptografického materiálu proběhne za využití asymetrické kryptografie, tak vlastní přenos dat běží šifrován symetrickou šifrou.
- Zajištění integrity přenášených dat na základně HMAC.
- TLS nezajišťuje důkaz pravosti přenášených dat, tj. neprovádí elektronické podepisování jednotlivých přenášený fragmentů či transakcí. Pokud



Obrázek 14.3 TLS je tvořen soustavou čtyř protokolů: RLP, HP, CCS a AP

aplikace vyžaduje podepisování např. transakcí, pak si to musí zajistit sama na aplikační vrstvě.

- Na počátku komunikace mezi klientem a serverem probíhá počáteční TLS dialog, kdy se server s klientem dohodnou na použitých kryptografických algoritmech, kryptografickém materiálu a provedou autentizaci. Mezi klientem a serverem se tak vytvoří TLS relace. Během této relace se aplikační data přenášejí zabezpečeně.
- Jelikož je vytvoření TLS relace poměrně náročné, je možné v rámci relace obnovit i další spojení. Při obnovování relace je úvodní dialog mezi klientem a serverem jednodušší a tím i méně náročný na výpočetní výkon, čas i kapacitu sítě
- Mezi klientem a serverem tak v rámce jedné relace můžeme mít několik spojení.
- Obnovené spojení v rámci téže relace využívá pozměněný kryptografický materiál.

Protokol TLS se skládá ze dvou dílčích protokolů (Obrázek 14.3):

1. **Protokol *Record Layer Protocol*** (RLP), který je z pohledu aplikačních protokolů onou „vrstvou TLS“ zabezpečující komunikaci. Protokol RLP bere data od aplikačních protokolů, šifruje je a počítá z nich HMAC. Druhý účastník spojení protokolem RLP dešifruje data, ověřuje HMAC a předává data aplikačnímu protokolu.

   Protokol RLP se nezabývá problémy jako je typ použitého šifrovacího algoritmu, stanovení šifrovacího klíče atd. Vše má připraveno protokolem HP v tzv. aktuální protokolové svitě a dalších parametrech zpracování.

2. ***Handshake protocol*** (HP) se z hlediska protokolu RLP tváří jako další aplikační protokol. Jeho pakety se balí do protokolu RLP. Protokolem HP se obě strany dohodnou na protokolové svitě (kryptografických algoritmech) i kryptografickém materiálu (kryptografické klíče a sdílená tajemství). Protokol HP se aktivuje bezprostředně po navázání TCP spojení a podle potřeby i během spojení. Protokol HP všechny kryptografické informace připraví nikoliv jako aktuální kryptografické parametry, ale jen jako připravované kryptografické parametry. Součástí protokolu HP jsou dále dva pomocné protokoly:

   - Protokol ***Change Cipher Specification Protocol*** (CCSP) je velice jednoduchý protokol. Poté, co protokol HP připraví nové kryptografické parametry pro činnost protokolu RLP, je třeba zkopírovat tyto připravené parametry na aktuální parametry a začít zabezpečovat podle nových parametrů. Protokol CCSP se skládá pouze z jediné zprávy „zkopíroval jsem připravované parametry na aktuální - nyní je šifrováno podle nových kryptografických parametrů“.
   - Dojde-li k jakémukoliv zádrhelu v komunikaci, pak je tu k dispozici ***Alert Protocol*** (AP), kterým jedna strana může signalizovat závadu druhé straně. AP lze s trochou nadsázky přirovnat k protokolu ICMP na vrstvě IP.

Připomeňme si, že protokol TCP je duplexním protokolem. Tj. vytváří dva komunikační kanály: Jeden pro tok dat z klienta na server a druhý pro tok dat opačným směrem ze serveru na klienta. Protokol TLS zabezpečuje každý kanál jiným kryptografickým materiálem.

- Pro tok dat z klienta na server TLS využívá:
  - Symetrickou šifru s šifrovacím klíčem označovaným client_write_key a případnými inicializačními vektory client_write_IV.
  - Sdílené tajemství client_write_MAC_secret pro zajištění integrity přenášených dat.
- Pro tok dat ze serveru na klienta TLS využívá:
  - Symetrickou šifru s šifrovacím klíčem označovaným server_write_key a případnými inicializačními vektory server_write_IV.
  - Sdílené tajemství server_write_MAC_secret pro zajištění integrity přenášených dat.

Komunikační kanál od klienta na server se často označuje jako client_write a komunikační kanál ze serveru na klienta jako server_write.

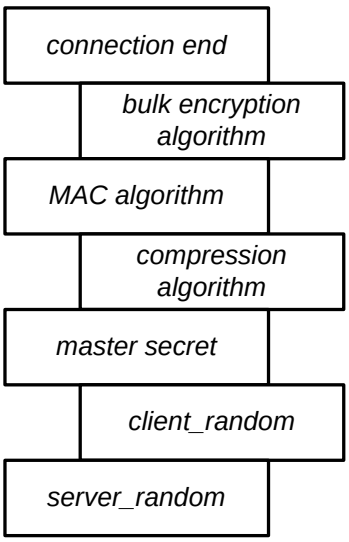
14.1 TLS relace a TLS spojení

Bezprostředně poté co klient naváže spojení protokolem TCP, startuje klient dialog protokolem HP. Cílem tohoto dialogu je vytvořit TLS relaci (*session*). Tj.:

- Dohodnout se na použitých kryptografických algoritmech. Klient nabízí sady algoritmů (tzv. protokolové svity) a server z nich volí.
- Mezi klientem a serverem vyměnit náhodná čísla client_random (generuje klient) a server_random (generuje server).
- Mezi klientem a serverem vyměnit certifikáty a kryptografické informace tak, aby se klient a server mohli vzájemně autentizovat. Autentizace je volitelná. V praxi se však používá autentizace alespoň serveru.
- Mezi klientem a serverem vyměnit kryptografické parametry tak, aby se klient a server mohli dohodnout na předběžném sdíleném tajemství (premaster secret), na jehož základě si pak oba mohou spočíst hlavní sdílené tajemství (master secret). Z hlavního sdíleného tajemství si následně klient i server odvodí kryptografický materiál. Délka předběžného tajemství je závislá na konkrétní volbě algoritmů; délka hlavního tajemství je vždy 48 bajtů.
- Umožnit klientu a serveru ověřit, že jejich protějšek spočetl stejné hlavní sdílené tajemství, tj. že jejich protějšek není podvržen.
- Poskytnout protokolu RLP kryptografické parametry (symetrické šifrovací klíče a sdílená tajemství pro výpočet HMAC).

Hlavní sdílené tajemství (*master secret*) je základem vztahu důvěry relace mezi klientem a serverem.

Pokud se mezi klientem a serverem zřizuje další spojení, pak je možné relaci obnovit. Při obnovení relace se vychází ze skutečnosti, že mezi klientem a



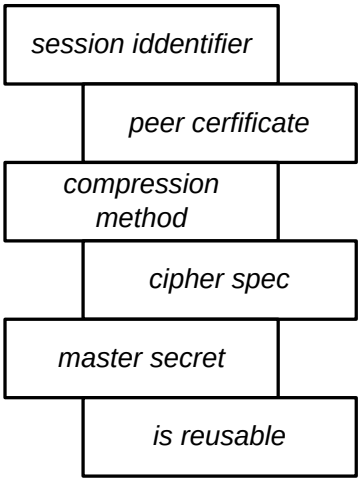
Obrázek 14.4 Nejdůležitější parametry spojení

serverem je již sdíleno hlavní tajemství. Není tedy třeba opakovat autentizaci (obě strany přece sdílí společné hlavní tajemství) ani vyměňovat mnohé kryptografické parametry.

Důsledkem je, že při obnovování si klient se serverem pouze vymění nová náhodná čísla client_random a server_random a přeskočí se body c. a d. Kryptografický materiál se pak následně spočte z hlavního sdíleného tajemství a těchto dvou nových náhodných čísel. Obnovení relace je tak výrazně rychlejší než zřízení nové relace. Říkáme, že v rámci relace zřizujeme další spojení (*connection*).

Můžeme také říci, že mezi klientem a serverem zřizujeme relaci (*session*), která může zahrnovat jedno nebo více spojení. Každá relace má svůj identifikátor relace. Tvůrci aplikací mohou považovat vytváření více spojení v rámci relace za bezpečnostně rizikové, pak relaci mohou označit jako neobnovitelnou, tj. při každém navázání spojení se zřizuje nová relace.

Obecně by relace neměla existovat déle než 24 hodin.



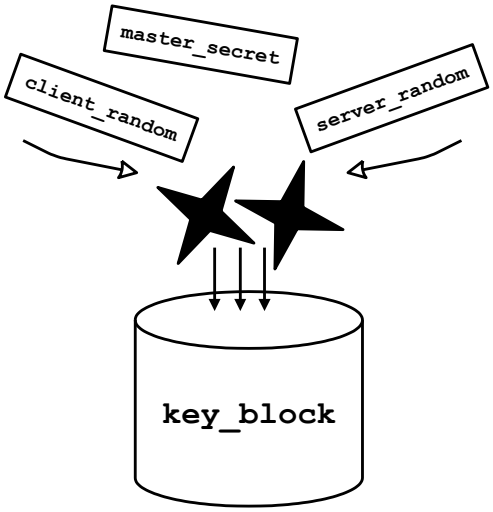
Obrázek 14.6 Nejdůležitější parametry relace

Klient i server ve svých strukturách udržují pro každou **relaci** následující parametry:

- Identifikační číslo relace (*session identifier*) - číslo jednoznačně identifikující relaci.
- Certifikát druhé strany (*peer certificate*).
- Komprimační algoritmus (*compression metod*) pro kompresi dat.
- Protokolovou svitu (*cipher spec*) specifikující symetrický šifrovací algoritmus a algoritmus pro výpočet HMAC.
- Hlavní sdílené tajemství (*master secret*), 48 bajtů známých pouze oběma účastníkům komunikace a utajované před ostatními uživateli sítě.
- Příznak, je-li možné relaci obnovovat (*is reusable*) nebo je nutné pokaždé navazovat novou relaci.

Pro jednotlivá **spojení** se pak udržují parametry:

- Typ konce spojení – server nebo klient (*connection end*).
- Symetrický šifrovací algoritmus (*bulk encryption algorithm*) včetně délky klíče a dalších parametrů.
- Algoritmus pro výpočet HMAC včetně parametrů.
- Komprimační algoritmus (*compression algorithm*).
- Hlavní sdílené tajemství (*master secret*); vždy dlouhé 48 bajtů.
- Náhodné číslo generované klientem (*client_random*).
- Náhodné číslo vygenerované serverem (*server_random*). Obě náhodná čísla jsou na počátku přenášena nezabezpečeně.

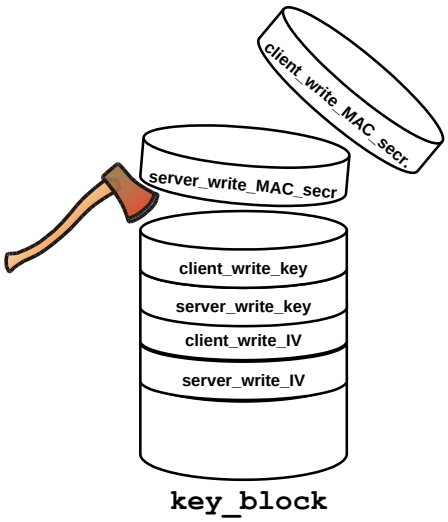


Obrázek 14.5 key_block

Otázkou je, jak se tajemství pro výpočet HMAC a symetrické šifrovací klíče stanoví. Nejprve se vytvoří blok klíčů (Obrázek 14.5) označovaný jako key_block. Tento blok se vytvoří pomocí pseudonáhodně funkce, do niž vstoupí: hlavní sdílené tajemství a náhodná čísla generovaná klientem a serverem.

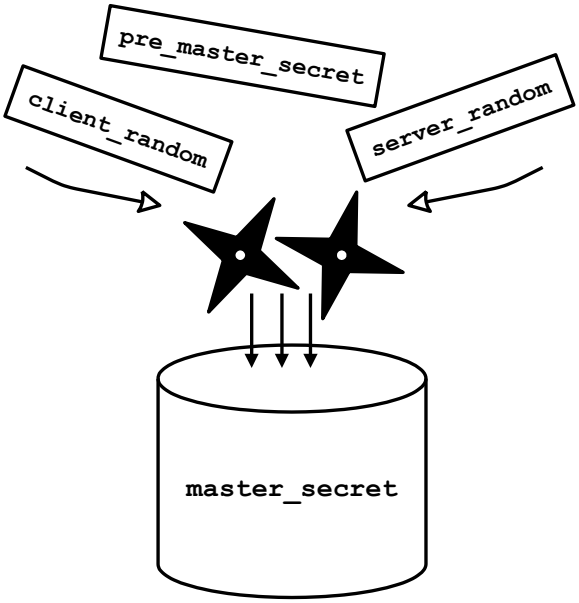
Z bloku klíčů se pak postupně odsekává:

- sdílené tajemství pro výpočet HMAC pro komunikaci od klienta na server (client_write_MAC_secret),



Obrázek 14.7 Z bloku klíčů se postupně odsekávají sdílená tajemství pro výpočet HMAC, symetrické šifrovací klíče a případně inicializační vektory

- sdílené tajemství pro výpočet HMAC pro komunikaci ze serveru na klienta (server_write_MAC_secret),



Obrázek 14.8 Stanovení hlavního sdíleného tajemství

- symetrický šifrovací klíč pro směr od klienta na server (client_write_key)
- symetrický šifrovací klíč pro směr ze serveru na klienta (server_write_key)
- inicializační vektory (clent_write_IV a server_write_IV). Inicializační vektory se však odsekávají jen v případě šifer, které inicializační vektory používají.

Odsekávání se provádí vždy v délce příslušného tajemství, resp. klíče, resp. IV.

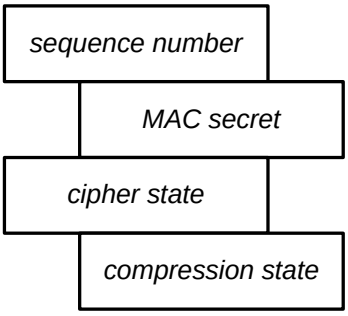
I v případě obnovování spojení je stanovení bloku klíčů stejné. I v tomto případě do výpočtu bloku klíčů vstupuje hlavní sdílené tajemství a náhodná čísla generovaná klientem a serverem, která jsou však vyměněna během obnovování spojení, tj. jedná se o nová náhodná čísla.

14.2 Autentizace

TLS umožňuje i komunikaci bez jakékoliv autentizace, tj. plně anonymní komunikaci. S tím se ale v praxi téměř nesetkáváme. Aplikace totiž téměř vždy vyžadují autentizaci serveru. TLS server, který se sám autentizuje, ale nevyžaduje autentizaci klientů se nazývá **anonymním TLS serverem**. Je to svým způsobem protimluv, protože server je autentizován a anonymními jsou klienti…

K autentizaci server i klient využívají certifikáty. V případě, že strana nezašle svůj certifikát druhé straně znamená to, že se nechce autentizovat.

Když začnete studovat protokol HP, asi vám nebude na první pohled jasné, kdy dochází k autentizaci serveru a kdy k autentizaci klienta.



Obrázek 14.9 Parametry zabezpečení toku dat

14.2.1 Autentizace serveru

Zejména autentizace serveru je v protokolu HP mírně zašmodrchaná. Důvodem je skutečnost, že současně s autentizací dochází mezi klientem a severem k výměně předběžného sdíleného tajemství. A nyní mohou nastat tři situace:

A. Certifikát serveru umožňuje jak autentizaci serveru, tak i bezpečný přenos předběžného sdíleného tajemství z klienta na server.

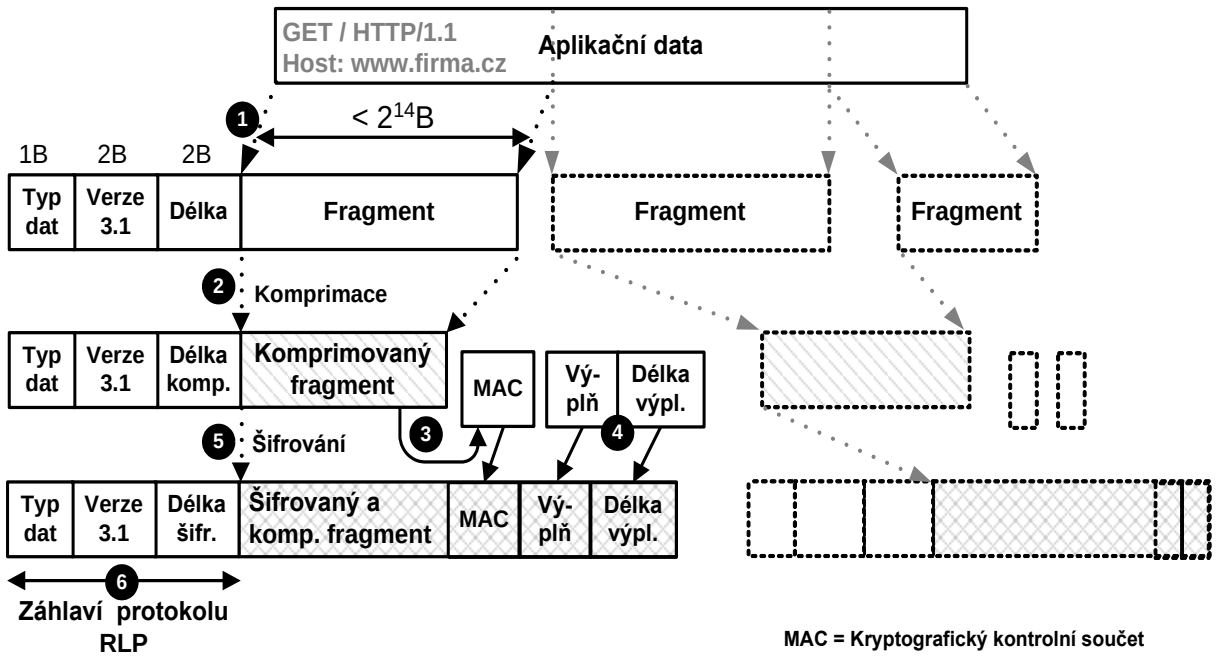
B. Certifikát serveru sice umožňuje autentizaci serveru, ale neumožňuje přenos předběžného sdíleného tajemství.

C. Server se autentizovat vůbec nechce, proto svůj certifikát vůbec nezasílá.

V případě A je situace velice jednoduchá. Serveru stačí zaslat na klienta svůj certifikát, ze kterého klient vyjme veřejný klíč, kterým šifruje předběžné sdílené tajemství. Výsledek pak zašle serveru. Server jej dešifruje svým soukromým klíčem. Tím server získá jak předběžné sdílené tajemství, tak se i autentizuje (nikdo jiný než server nemá příslušný dešifrovací soukromý klíč).

V případě C musí server generovat dočasná párová data a dočasný veřejný klíč zaslat klientovi aby jej využil k zabezpečení předběžného tajemství.

Nejsložitějším je případ B, kdy server rovněž musí generovat dočasná párová data, ale ta musí nějakým způsobem navíc svázat se svým certifikátem. Tou vazbou je již zmíněný elektronický podpis TLS. Server elektronicky podepíše dočasný veřejný klíč, který zasílá klientovi.



Obrázek 14.10 Zabezpečení dat protokolem RLP pro případ blokové šifry Záhlaví RLP se skládá ze tří položek.

V terminologii protokolu HP (viz dále) server zasílá svůj certifikát klientu ve zprávě Certificate. Případný dočasný veřejný klíč včetně jeho případného podpisu pak server zasílá ve zprávě ServerKeyExchange.

14.2.2 Autentizace klikenta

Autentizace klienta je výrazně jednodušší. Pokud se chce klient autentizovat, pak klient zašle za sebou serveru zprávy Certificate a CertificateVerify:

- Ve zprávě Certificate zašle svůj certifikát
- Ve zprávě CertificateVerify zašle elektronický podpis (v smyslu TLS) z předchozí komunikace protokolem HP. Ta též obsahovala náhodná čísla generovaná klientem i serverem. Tj. klient posílá elektronický podpis z řetězce, jehož části jsou náhodná čísla client_random a server_random čímž omezuje replay attack.

Server následně provede verifikaci tohoto elektronického podpisu a autentizace je hotova (nikdo jiný než klient nemá k dispozici příslušný soukromý klíč, kterým byl vytvořen elektronický podpis TLS).

14.3 Předběžné a hlavní sdílená tajemství

Pokud server poskytl veřejný klíč umožňující šifrování dat, klient generuje předběžné tajemství jako náhodné číslo zřetězené s datem a časem. Takto vygenerované předběžné tajemství šifruje veřejným klíčem serveru a zašle jej serveru. Server si jej dešifruje svým soukromým klíčem a obě strany znají předběžné sdílené tajemství.

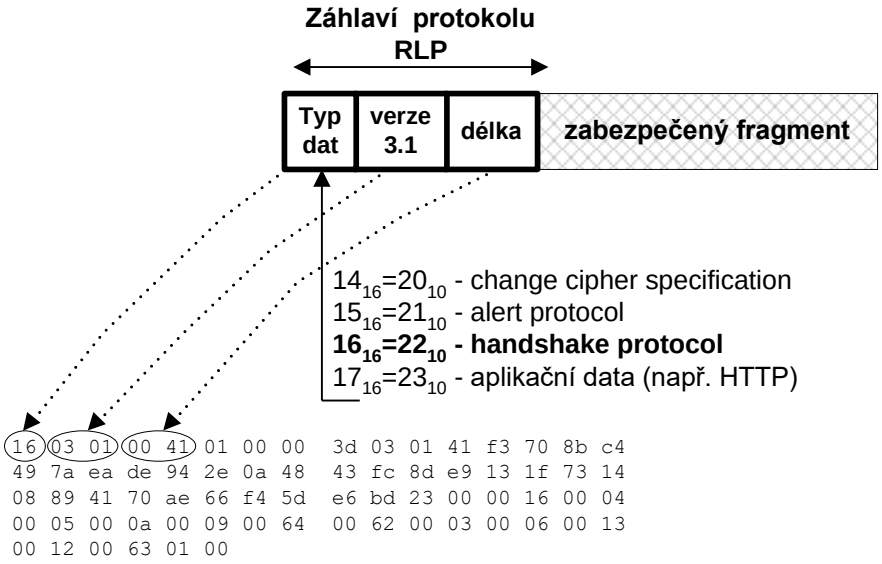
V případě, že server zaslal své veřejné Diffie-Hellmanovo číslo, pak klient odpovídá svým veřejným Diffie-Hellmanovým číslem. Sdílené tajemství si obě strany následně spočtou Diffie-Hellmanovým algoritmem ze svých párových dat a veřejného čísla protějšku. (Pakliže klientův certifikát již veřejné Diffie-Hellmanovo číslo obsahoval, pak klient serveru pochopitelně další veřejné Diffie-Hellmanovo číslo zasílat nemusí.)

Jelikož pro stanovení předběžného tajemství mohou být využity různé metody, tak i délka předběžného tajemství může být různá. Proto z předběžného tajemství se opět pseudonáhodnou funkcí spočte hlavní tajemství (Obrázek 14.8), které je vždy dlouhé 48 bajtů. Do výpočtu hlavního tajemství se opět přimelou náhodná čísla `client_random` a `server_random`.

Po ustanovení hlavního tajemství se předběžné tajemství smaže!

14.4 Record Layer Protocol (RLP)

Protokol RLP přebírá data od aplikačních protokolů a zabezpečí je pro přenos „nebezpečnou“ sítí. Na druhém konci sítě pak protokol RLP data dešifruje, verifikuje příslušné HMAC a předá je aplikačním protokolů



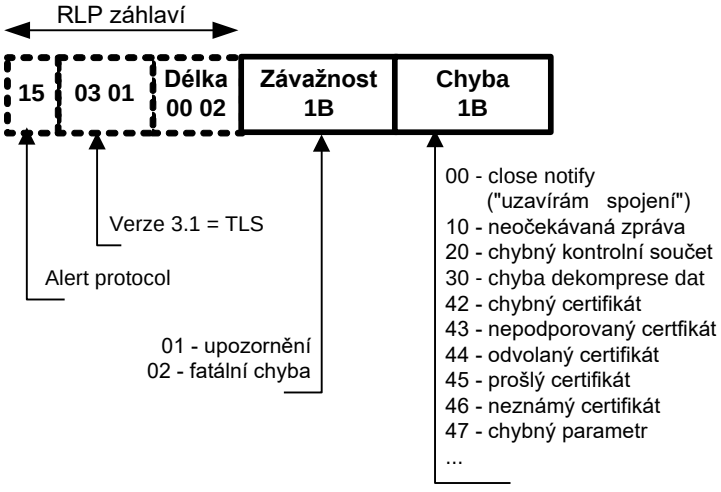
Obrázek 14.11 Fragment protokolu RLP

Protokol RLP pro každý směr komunikace (`server_write` i `client_write`) udržuje, a po zpracování každého fragmentu dat aktualizuje, následující parametry:

- Pořadové číslo zpracovaného fragmentu (sequence number). Pořadové číslo se vždy počítá od nuly a nesmí překročit 264-1.
- HMAC fragmentu (MAC secret).
- Stav symetrické šifry (cipher state). Např. pro blokové šifry v CBC módu je na počátku naplněn inicializačními vektory.
- Stav kompresního algoritmu (compresssion state).

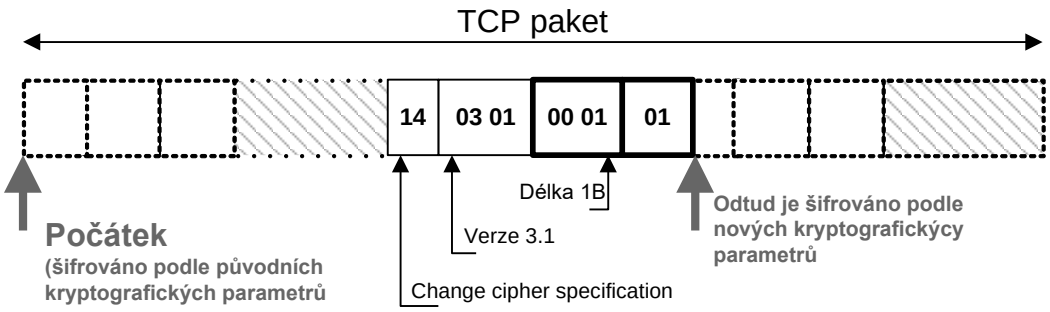
Protokol RLP před tím, než data předá vrstvě TCP provede (sledujte Obrázek 14.10) :

- Rozsekání aplikačních dat na fragmenty o délce menší než 214 bajtů.
- Komprimaci dat, pokud se na ní dohodli klient se serverem (pomocí HP protokolu). Komprimovaný fragment se pomyslně předsadí záhlavím RLP protokolu. Komprimovaný fragment včetně svého záhlaví následně vstupuje do výpočtu HMAC.
- Výpočet HMAC komprimovaného fragmentu algoritmem, na kterém se dohodli klient a server protokolem HP. Do výpočtu HMAC vstupují:
  - Tajemství pro výpočet HMAC (*_write_MAC_secret)
  - Pořadové číslo zpracovávaného fragmentu
  - Data komprimovaného fragmentu včetně jeho záhlaví.
- V případě blokových šifer doplnění fragmentu o výplň na délku, která je násobkem délky šifrovacího bloku. Výplň je následována jedním bajtem obsahujícím délku výplně. V případě proudových šifer se žádná výplň nepřidává. Žádná výplň se rovněž nepřidává, pokud se fragmenty nešifrují (např. úvodní dialog protokolem HP).
- Šifrování fragmentu.
- Doplnění hlavičky RLP protokolu před datový fragment.



Obrázek 14.12 Zpráva protokolu AP se vkládá do RLP fragmentu

Záhlaví RLP se skládá ze tří položek (Obrázek 14.11).



Obrázek 14.13 práva protokolu Change Cipher Specification

1. Typu přenášených dat specifikujícího, jedná-li se o data aplikačního protokolu (např. HTTP) nebo o služební data vrstvy TLS. Typ dat je dlouhý 1 B a nabývá hodnot: 20 pro CCSP, 21 pro AP, 22 pro HP a 23 pro aplikační data.
2. Verze, která se skládá ze dvou bajtů: první bajt obsahuje verzi a druhý nějaké jemnější dělení. Prakticky se však můžeme setkat s hodnotami:
    - 2.0 pro SSL verze 2;
    - 3.0 pro SSL verze 3. a
    - 3.1 pro TLS verze 1.
3. Délky fragmentu po jeho případné komprimaci a šifrování. Pole délka fragmentu je dlouhá 2 B (= 16 b). Tj. fragment by teoreticky mohl být dlouhý až 216 B. Díky kompresi může být efektivní délka dat skutečně přenesených ve fragmentu větší.

14.5 Alert protocol

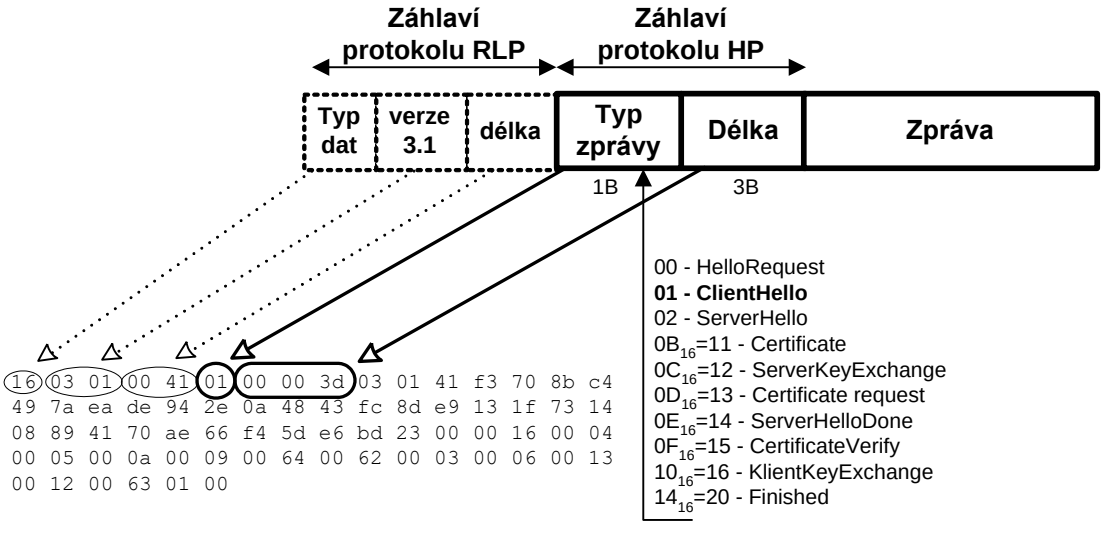
Alert protocol je jednoduchý protokol, kterým se účastníci komunikace vzájemně informují o chybách v TLS komunikaci. Protokol AP má jediný typ zprávy, který obsahuje dva bajty (Obrázek 14.12). První indikuje závažnost chyby a druhý bajt specifikuje konkrétní chybu. Rozeznávají se dvě úrovně závažnosti:

1. Upozornění, po kterém může komunikace dále pokračovat.
2. Fatální chyba, po které se komunikace ukončuje.

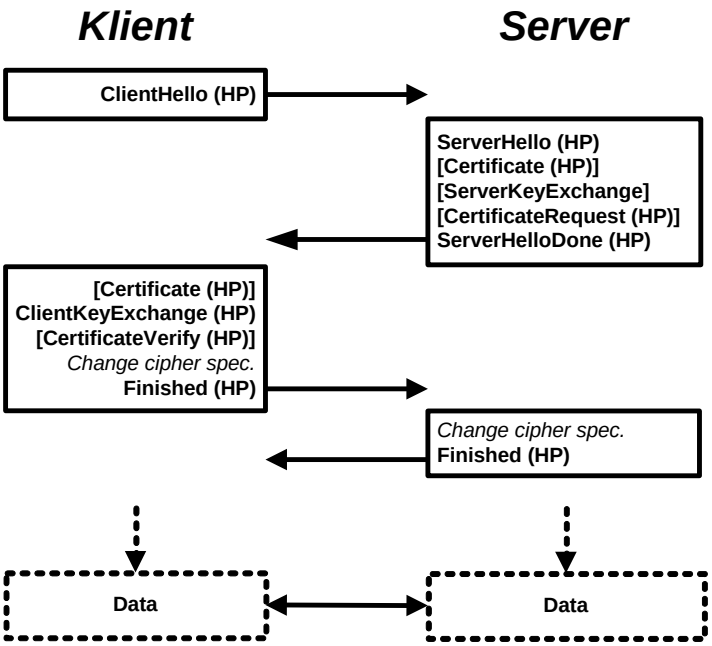
14.6 Change Cipher Specification Protocol (CCSP)

CCSP je protokol skládající se pouze z jedné zprávy (zprávy ChangeCipherSpecification). Tato zpráva signalizuje, že došlo ke zkopírování připravovaných kryptografických parametrů na aktuální kryptografické parametry. Další fragment protokolu RLP již tedy bude zabezpečen dle nových kryptografických parametrů. Veškerá data, která následují za touto zprávou, jsou tak zabezpečena za využití nového šifrovacího klíče, nového tajemství pro výpočet HMAC atd.

Slovně bychom mohli zprávu ChangeCipherSpecification vyjádřit jako „změna šifrovacího klíče“. Jen je třeba dodat, že tuto změnu provádí klient a server na sobě nezávisle, tj. jedna strana provede změnu



Obrázek 14.14 Handshake Protocol



Obrázek 14.15 Zřízení nové relace protokolem HP

ve svém směru komunikace a druhá např. až za okamžik - za svou zprávou ChangeCipherSpecification.

TCP segment může obsahovat zprávu ChangeCipherSpecification uprostřed, tj. část RLP-fragmentu předcházející tuto zprávu je šifrována jinak než část RLP-fragmentu za touto zprávou.

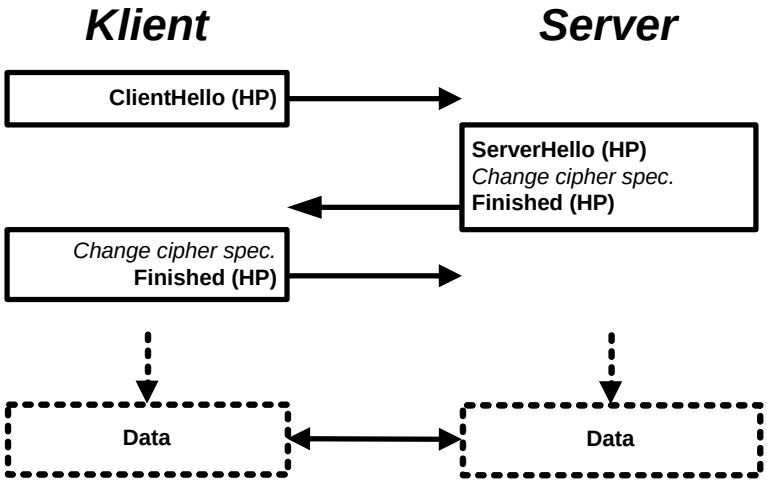
14.7 Handshake protocol (HP)

V protokolu HP komunikuje klient se serverem pomocí tzv. zpráv. Zprávy protokolu HP se vkládají do RLP fragmentů. Zprávy se skládají ze záhlaví a těla zprávy (Obrázek 14.14). Na rozdíl od protokolu RLP je zabezpečena celá zpráva včetně záhlaví.

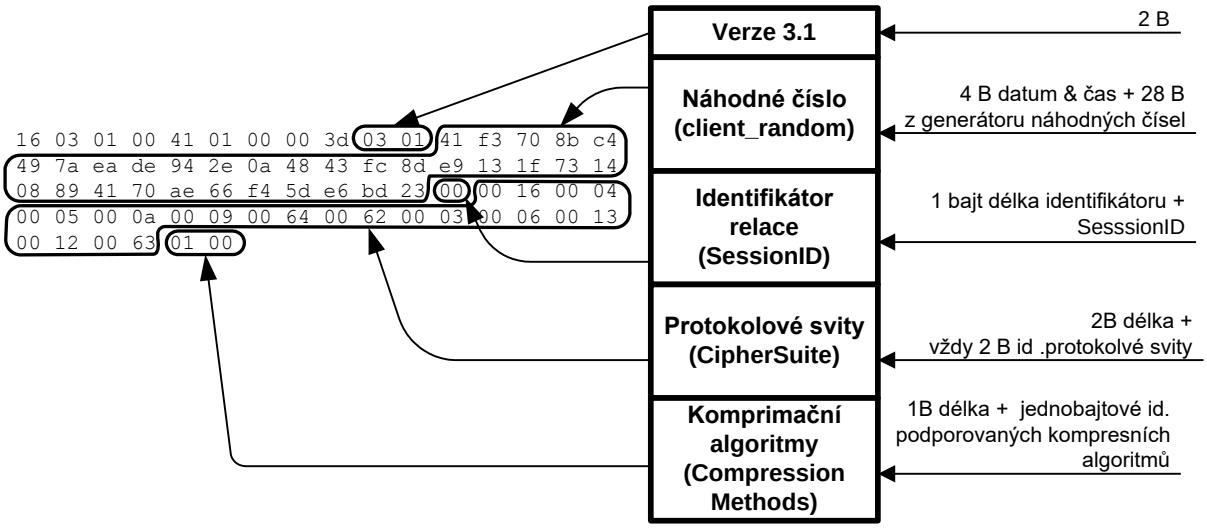
.

Každá zpráva protokolu HP má pak vlastní strukturu. Rozeznávají se tři případy komunikace protokolem HP:

1. Navazování nové relace, tj. server přiděluje nový identifikátor relace a veškeré kryptografické algoritmy a veškerý kryptografický materiál si obě strany dohadují od počátku.
2. Obnovení relace (použije se původní identifikátor relace). Obnovením relace vznikne další spojení téže relace. Tento případ je v podstatě zjednodušením předchozího případu,. tj. používají se jen některé zprávy.
3. Obnovení kryptografického materiálu v rámci běžícího spojení po uplynutí stanoveného časového intervalu. Tento případ je sice obdobou



Obrázek 14.16 Obnovení relace protokolem HP



Obrázek 14.17 Zpráva ClientHello

předchozího případu, dialog však probíhá zabezpečeně dle stávajících kryptografických parametrů.

14.7.1 Zřízení nové relace

Při zřizování i obnovování relace začíná dialog vždy klient a to zprávou ClientHello (Obrázek 14.15). Zpráva ClientHello obsahuje nevyplněné identifikační číslo relace (SessionID). Identifikační číslo relace přiděluje server ve své odpovědi (ve zprávě ServerHello). Pokud si server nepřeje obnovování relace, pak identifikátor relace může klientovi zatajit, tj. jej vůbec klientovi neposlat.

Zprávy ClientHello a ServerHello obsahují náhodná čísla client_random a server_random vstupující do procesu výpočtu sdíleného tajemství. Dále klient ve zprávě ClientHello nabídne podporované kryptografické algoritmy a server si ve zprávě ServerHello z nich vybere.

Pokud se server chce autetnizovat, pošle klientovi svůj certifikát (včetně případného řetězce certifikátů) ve zprávě Certificate. Pokud server svůj certifikát nezašle, nebo se zaslaný certifikát serveru nehodí pro zabezpečení přenosu předběžného sdíleného tajemství z klienta na server, pak server vygeneruje dočasná párová data a veřejnou část těchto dat odešle ve zprávě ServerkeyExchange. Pokud server podporuje autentizaci klienta, pak server ve zprávě CerificateRequest předá klientovi seznam jedinečných jmen (*Distinguished Name*) pro server důvěryhodných certifikačních autorit klientů. Nakonec server odešle prázdnou zprávou ServerHelloDone, kterou signalizuje klientu, že nyní je řada na něm, tj. očekává reakci klienta.

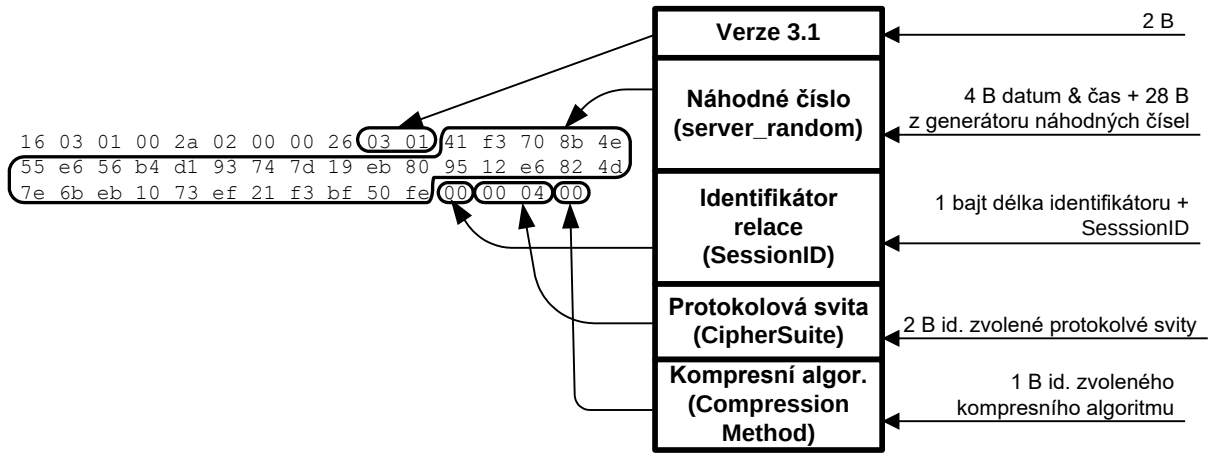
V případě že klient chce, na server přistupovat neanonymně, odešle serveru svůj certifikát (resp. řetězec certifikátů). Dále následuje zcela klíčová zpráva ClientKeyExchange. Tato zpráva obsahuje veřejným klíčem z certifikátu serveru šifrované předběžné tajemství[*)], ze kterého se počítá hlavní tajemství (Obrázek 14.8). Tím se také provede autentizace serveru. Podvržený server, který by k veřejnému klíči nevlastnil odpovídající soukromý klíč, nemůže získat správné předběžné tajemství (nedokáže jej dešifrovat).

V případě, že klient přistupuje neanonymně na server, pak musí prokázat svou totožnost, tj. identifikovat se. Klient odeslal svůj certifikát ve zprávě Certificate, avšak nyní ještě musí prokázat svou totožnost pomocí zprávy CertificateVerify. Svou totožnost prokáže pomocí elektronického podpisu počítaného z obsahu všech předchozích zpráv až od zprávy ClientHello. Tento elektronický podpis je schopen vytvořit jen klient mající soukromý klíč k certifikátu zaslaném ve zprávě Certificate. Server si pak ověří totožnost klienta na základě ověření (verifikace) tohoto elektronického podpisu.

Nyní již obě strany sdílí společné hlavní tajemství, ze kterého jsou schopny odvodit veškerý kryptografický materiál. Klient tak může protokolem *Change Cipher Specification* signalizovat, že „přešel na nové šifrovací klíče“. Následuje zpráva Finished odeslaná klientem na server. Ta je již zabezpečena. Zpráva Finished obsahuje důkaz, že odesilatel má správně sestaveno společné sdílené tajemství.

Na závěr i server signalizuje protokolem *Change Cipher Specification* „přechod na nové šifrovací klíče“ (ve směru komunikace ze serveru na klienta).

[*)] O modifikaci pro Diffie-Hellmanův algoritmus jsme se již zmínili.



Obrázek 14.18 ServerHello

Poslední zprávou je pak a zpráva Finished odeslaná serverem, která opět nese důkaz o správném sestavení hlavního sdíleného tajemství serverem. Navazování spojení končí. Dále již běží přenos vlastních aplikačních dat, který je šifrován. Každý RLP fragment obsahuje též HMAC zabezpečující integritu přenášených dat (už zprávy Finished byly šifrovány).

Zprávy Certificate nesly certifikát serveru nebo klienta včetně případného řetězce certifikátů až k důvěryhodné kotvě. Přítomnost důvěryhodné kotvy může být nejvýše informativní. Důvěryhodnost důvěryhodné kotvy musí být ověřena jinou cestou.

Zajímavé je, že zprávy Certificate nesou již sestavený řetězec certifikátů počínající certifikátem serveru, resp. klienta. Je to rozdíl např. od protokolu CMS popisovaného v kap. 24. Zprávy protokolu CMS nesou totiž množiny certifikátu a je na ověřovateli zprávy, aby si v množině certifikátů nalezl příslušný řetězec certifikátů.

14.7.2 Obnovení relace

Jednodušším případem komunikace mezi klientem a serverem je obnovení již dříve existující relace (Obrázek 14.16). Kdy se s takovým případem setkáváme? Obnovením se většinou nemyslí (i když se to nevylučuje) obnovení spojení např. po závadě na komunikačních linkách - to je záležitost protokolu nižší vrstvy (v našem případě protokolu TCP). Spíše se zde bude jednat o trochu jiný případ. Představte si, že si prohlížíte webové stránky na HTTPS serveru. Při vydání požadavku na stažení webové stránky musí být navázána relace. Pro stažení další stránky je pak možné relaci obnovit a tím zrychlit komunikaci. Některé webové servery vyžadují navazování spojení i pro jednotlivé objekty webové stránky, pak obnovování TLS relací oceníme ještě více.

Při obnovování spojení dialog opět začíná klient zprávou ClientHello. Zpráva ClientHello obsahuje náhodné číslo client_random vstupující do výpočtu bloku klíčů nového spojení. Zpráva ClientHello obsahuje rovněž identifikátor relace, kterou si přeje klient obnovit. Dále zpráva ClientHello obsahuje seznam kryptografických algoritmů podporovaných klientem (seznam by měl obsahovat i algoritmy použité v předchozím spojení).

Server odpovídá zprávou ServerHello obsahující náhodné číslo server_random též vstupující do výpočtu bloku klíčů nového spojení. Zpráva ServerHello obsahuje opět identifikátor obnovované relace a zejména kryptografické algoritmy, které si server vybral z těch, které klient podporuje. Další kroky, které jsme poznali v případě navázání spojení se již přeskočí. Obě strany znají z minula hlavní sdílené tajemství relace a nyní získaly i nová náhodná čísla client_random a server_random. Mohou si tedy spočíst nový blok klíčů (Obrázek 14.7). Protokolem *Change Cipher Specification* přejdou do zabezpečené komunikace, ve které si zprávou Finished ověří, že mají správné hlavní sdílené tajemství. A pokud je vše v pořádku, může se začít s přenosem dat.

Nyní si rozebereme jednotlivé zprávy podrobněji. Pro ilustraci použijeme zprávy, které jsem opět odchytil na LAN pomocí programu Ethereal (nebudu však uvádět již kompletní výpis, ale jen části týkající se protokolu TLS).

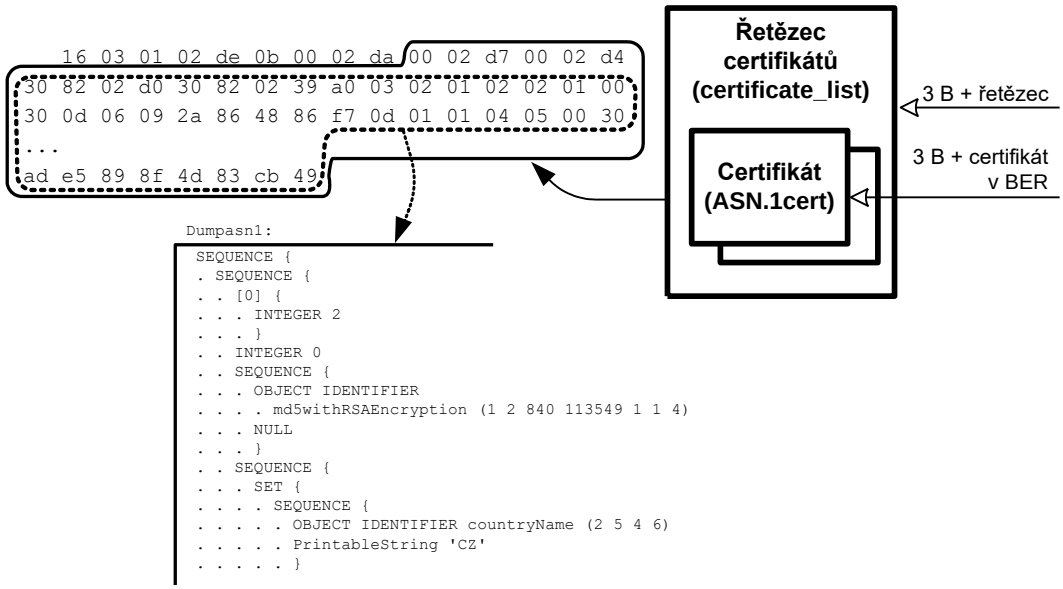
14.7.3 Zpráva ClientHello

Zpráva `ClientHello` je znázorněna na obr. Obrázek 14.17. Na obrázku vlevo je TLS fragment odchycený na síti (např. programem Ethereal). Vpravo je pak znázorněna struktura zprávy.

Zpráva je uvozena RLP záhlavím: 16 03 01 00 41 (zpráva protokolu HP, verze 3.1, délka $41_{16}=65_{10}$)

Dále následuje záhlaví zprávy `ClientHello`: 01 00 00 3d (01 = zpráva `ClientHello`, délka zprávy je $3d_{16}=61_{10}$)

Obsah zprávy `ClientHello` je tvořen:



Obrázek 14.19 Zpráva Certificate

1. Verzí protokolu TLS (pro verzi 1 je hodnota 3.1) dlouhou 2B. Verze protokolu u zprávy ClentHelo a ServerHelou je důležitá pro zpětnou kompatibilitu o čemž se ještě zmíníme.
2. 32 B dlouhým náhodným číslem (client_random) generovaným klientem. Toto náhodné číslo se skládá ze dvou částí:
    - První 4 B tvoří datum a čas v Unixovém tvaru (viz kap. 21.1.5).
    - 28 B výstupu z generátoru náhodných čísel. V minulosti se vyskytly právě námitky proti „náhodnosti“ tohoto náhodného čísla.
3. Identifikátorem relace (SessionID) uvozený jedním bajtem délky relace. Při prvním navazování nového spojení je tato položka nulové délky, tj. prázdná, protože SessionID určuje až server ve zprávě ServerHello. V případě, že se spojení obnovuje, pak toto pole obsahuje číslo obnovované relace.
4. Seznam tzv. protokolových svit uvozený dvěma bajty délky seznamu. Protokolové svity jsou dvojbajtové řetězce identifikující asymetrický protokol, symetrický protokol a protokol pro výpočet otisku. Klient v této zprávě popisuje všechny jím podporované protokolové svity. Říká: "Já podporuji následující svity. Servere, vyber si z nich jednu a tu mi pošli ve zprávě ServerHello". První dva bajty vyjadřují délku řetězce předávaných protokolových svit. Seznam podporovaných kompresních algoritmů uvozený jedním bajtem délky seznamu. Náš klient podporuje následující kompresní algoritmy:

```
00 – bez komprese
```

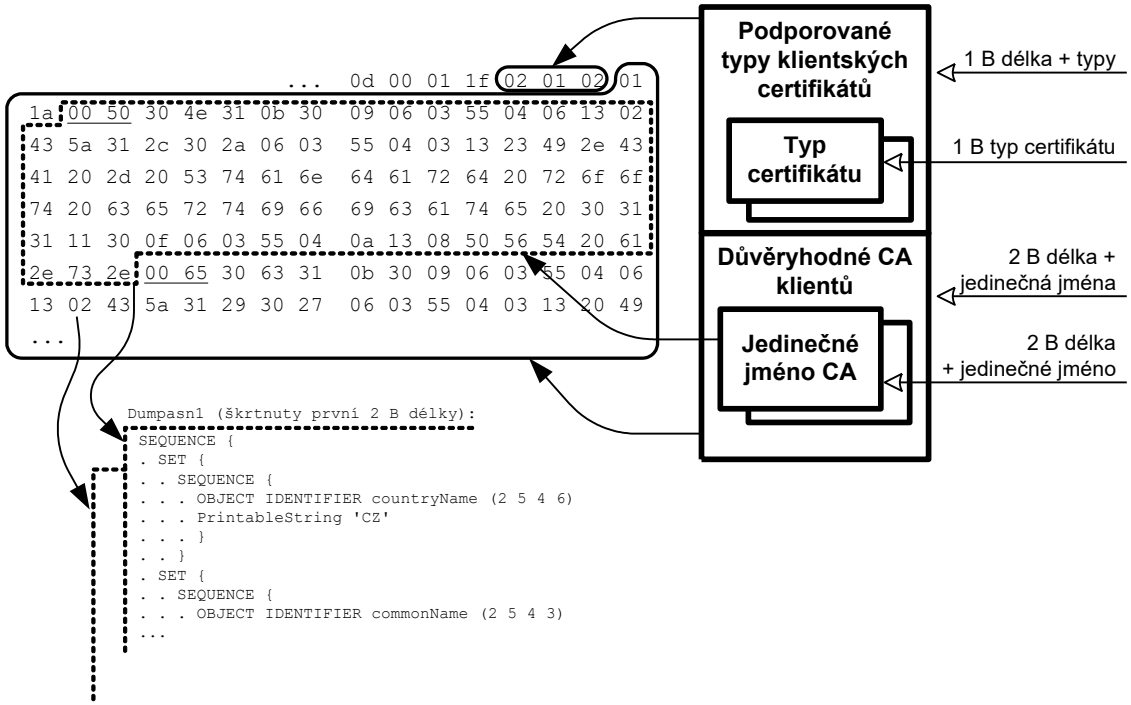
14.7.4 Zpráva ServerHello

Zpráva `ServerHello` (Obrázek 14.18) je obdobou zprávy `ClientHello`. Zpráva ServerHello obsahuje:

1. Verzi protokolu TLS (3.1) dlouhou 2B.
2. Náhodné číslo server_random generované serverem, které má stejnou strukturu jako číslo client_random.
3. Identifikátor relace (SessionID) uvozený jedním bajtem délky relace, který je v případě obnovování relace zkopírován ze zprávy ClientHello. V našem případě server identifikátor relace neuvádí, tj. server nechce, aby relace byla obnovována. Jedná se tedy o neobnovitelnou relaci.
4. Zvolenou protokolovou svitu 00 04 (TLS_RSA_WITH_RC4_128_MD5).
5. Zvolený kompresní algoritmus 00 (bez komprese)

14.7.5 Zpráva Certificate

Zpráva Certificate obsahuje řetězec certifikátů, který nemusí končit důvěryhodnou kotvou, protože ta musí být distribuován jinou (důvěryhodnou) cestou. Řetězec certifikátů je uvozen třemi bajty délky řetězce.



Obrázek 14.20 Zpráva CertificateRequest

Řetězec certifikátů začíná certifikátem serveru nebo klienta v závislosti na tom, kdo zprávu odesílá. Každý certifikát v řetězci je opět uvozen třemi bajty délky certifikátu. Na obr. Obrázek 14.19 je řetězec tvořen jen jedním certifikátem, který předchází tři bajty délky řetězce (00 02 d7) a třemi bajty délky certifikátu (00 02 d4). Jednotlivé certifikáty jsou kódovány v DER, takže si jejich obsah můžeme vypsat např. oblíbeným programem dumpasn1.

Stejnou zprávou Certificate případně odesílá i klient svůj certifikát na server.

14.7.6 Zpráva CertificateRequest

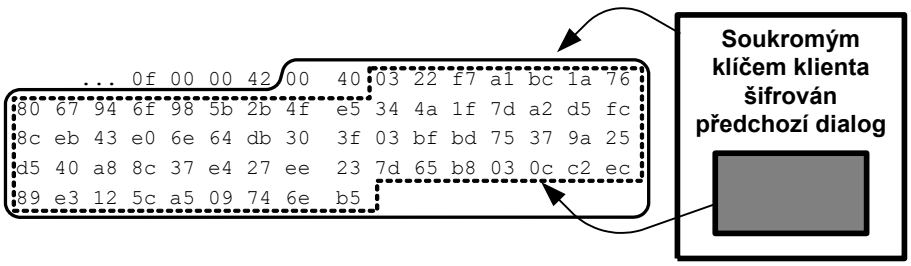
Touto zprávou (Obrázek 14.20) žádá server klienta o jeho certifikát. Součástí zprávy je seznam podporovaných typů certifikátů a seznam certifikačních autorit, kterým server důvěřuje.

Tato zpráva se skládá ze dvou polí:

1. Seznamem typů podporovaných certifikátů předcházeným jedním bajtem délky seznamu. Náš server podporuje následující typy klientských certifikátů

    - 01 = rsa_sign
    - 02 = dss_sign

2. Seznam jedinečných jména (*Distinguished Name*) klientských certifikačních autorit, kterým server důvěřuje. Toto pole je uvozenou dvojbajtovou délkou. Každé jedinečné jméno je pak také předcházeno dvojbajtovou délkou (na orázku podtrženo).



Obrázek 14.21 CertificateVerify

Na zprávu CertificateRequest klient odpoví zprávou Certificate, která má stejný formát jako zpráva Certificate serveru. Pokud klient požadovaný certifikát nemá, pak to ještě nemusí způsobit fatální chybu, protože server může podporovat zároveň autentizaci certifikátem i anonymní přístup.

Aplikační programy (např. webové prohlížeče) často uživatelům zobrazí seznam uživatelových certifikátů vydaných certifikačními autoritami důvěryhodnými pro server. Uživatel si pak vybere ze seznamu a software zajistí zaslání tohoto zvoleného certifikátu na server. Častým dotazem uživatelů je proč se jim seznam certifikátů zobrazuje prázdný, když oni mají vydané osobní certifikáty. Odpověď je jednoduchá: mají vydané osobní certifikáty od jiných CA než těch, kterým tento server důvěřuje. Tj. jiných něž těch jejichž jedinečná jména zaslal ve zprávě CertificateRequest. Pikantní na tom je, že chyba může být i na serveru, protože např. správce serveru nezařadil do příslušného konfiguračního souboru serveru nový certifikát klientské CA.

14.7.7 Zpráva ServerHelloDone

Tato zpráva protokolu HP je prázdná - neobsahuje žádná data (Obrázek 14.22). Vždy se skládá jen ze záhlaví:

0E 00 00 00

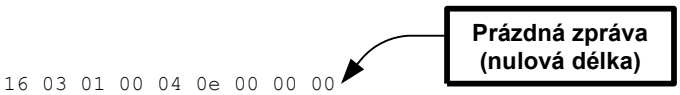
kde:

- 0E je typ zprávy (ServerHelloDone)
- 00 00 00 je délka zprávy (= nula, tj. - neobsahuje žádná data).

14.7.8 Zpráva ClientKeyExchange

V případě, že server ve svém certifikátu nebo ve zprávě ServerKeyExchange poskytl klientu veřejný klíč, pak:

- Klient generuje předběžné sdílené tajemství jako 46-bajtové náhodné číslo.



Obrázek 14.22 Zpráva ServerHelloDone

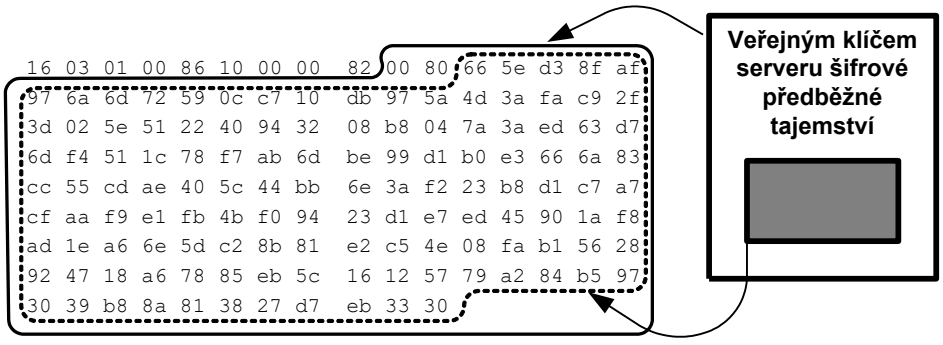
- Předběžné tajemství šifruje veřejným klíčem serveru.
- Výsledek vloží do zprávy ClientKeyExchange (Obrázek 14.23).

Zpráva ClientKeyExchange tak v případě veřejného klíče (např. algoritmu RSA) obsahuje veřejným klíčem serveru šifrované předběžné tajemství. Server si pak předběžné sdílené tajemství dešifruje svým soukromým klíčem. Poté klient i server znají předběžné sdílené tajemství. To se následně převede na hlavní sdílené tajemství (Obrázek 14.8). Po stanovení hlavního sdíleného tajemství by aplikace měly smazat předběžné sdílené tajemství, aby nemohlo později dojít k jeho případné kompromitaci.

V případě, že server neposkytl veřejný klíč, ale veřejné Diffie-Hellmanovo číslo, pak mohou nastat dva případy:

- Klient se neautentizuje vůbec nebo se autentizuje certifikátem, který neobsahuje veřejné Diffie-Hellmanovo číslo (např. klient se autetnizuje certifikátem veřejného klíče algoritmu RSA). Pak klient generuje dočasná Diffie-Hellmanova čísla. A do zprávy ClientKeyExchange vloží své dočasné veřejné Diffie-Helmanovo číslo.
- Klient se autentizuje pomocí certifikátu veřejného Diffie-Hellmanova čísla. Pak sice také pošle zprávu ClientKeyExchange, ale prázdnou.

Předběžné sdílené tajemství se pak spočte z Diffie-Hellmanových čísel. V tomto případě klient nemusí odesílat zprávu CertificateVerify, protože autentizace klienta je na základě držení jeho soukromého Diffie-Hellmanova čísla.

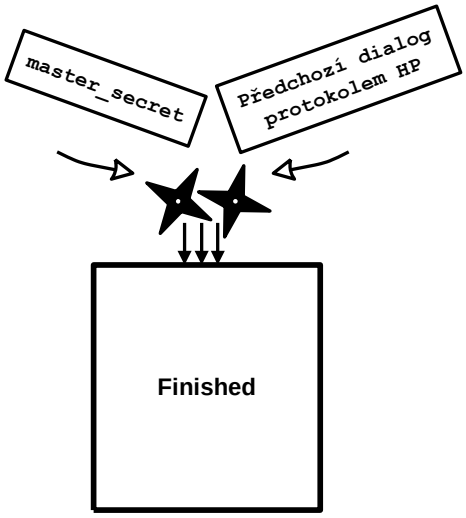


Obrázek 14.23 Zpráva ClientKeyExchange

14.7.9 Zpráva CertificateVerify

V případě, že klient ve zprávě Certificate odeslal certifikát umožňující elektronický podpis (elektronický podpis ve smyslu TLS), provede klient svou autentizaci pomocí zprávy CertificateVerify .

Zpráva obsahuje elektronický podpis ze všech zpráv vyměněných mezi klientem a serverem od zprávy ClientHello, nezapočítávaje v to tuto zprávu (Obrázek 14.21).



Obrázek 14.24 Zpráva Finished

14.7.10 Zpráva Finished

Server i klient ukončuje dialog v protokolu HP zprávou Finished, jež je již šifrována (následuje po zprávě ChangeCipherSpecification).

Obsahem datové části zprávy Finished je HMAC z hlavního sdíleného tajemství a ze všech předchozích zpráv od zprávy ClientHello (nezapočítávaje tuto zprávu). Zprávu Finished je tedy schopen sestavit jen ten, kdo zná hlavní sdílené tajemství. Zpráva Finished je tedy i důkazem, že druhá strana má k dispozici hlavní sdílené tajemství.

Ze záhlaví RLP fragmentu předcházející vlastní zprávu Finished jsme schopni poznat pouze, že se jedná o HP protokol (16) verzi (03 01) a délku zprávy, dále už je vše šifrováno. Nemá tedy smysl se pitvat ve zprávách Finished odchycených na síti.

14.7.11 Zpráva ServerKeyExchange

V případě, že server certifikát neposílá (jedná se o zcela anonymní komunikaci) nebo server sice certifikát posílá, ale ten nelze využít k zabezpečení přenosu předběžného sdíleného tajemství od klienta na server, pak server generuje dočasná párová data a jejich veřejnou část odešle zprávou ServerKeyExchange klientu (Obrázek 14.25)

Zpráva ServerKeyExchange se skládá ze dvou částí:

1. Veřejné části párových dat.
2. Elektronického podpisu veřejné části párových dat. Tato část slouží ke svázání certifikátu serveru s dočasnými párovými daty. Je to důkaz, že dočasná párová data byla generována serverem, který předložil svůj certifikát v předchozí zprávě Certificate. V případě, že se server neodeslal svůj certifikát (plně anonymní komunikace), je tato část prázdná.

Obrázek 14.25 Zpráva ServerKeyExchange

14.7.12 Zpráva HelloRequest

Zpráva HelloRequest je prázdná, málo běžná zpráva, kterou server upozorňuje klienta, že je na čase, aby klient odeslal zprávu ClientHello za účelem obnovení kryptografického materiálu. Jedině klient totiž může odeslat zprávu ClientHello, kterou HP protokol začíná dialog vedoucí k ustanovení nových šifrovacích klíčů.

14.8 Zpětná kompatibilita

Pokud si v konfiguraci vašeho prohlížeče zatrhnete, že chcete používat nejenom protokol TLS, ale i protokol SSL a to nejenom verze 3, ale i dokonce verze 2, pak při analýze programem Ethereal nachytané komunikace budete překvapeni, že klient je schopen se se serverem dohodnout na nejvyšší možné verzi. Klient v takovém případě totiž navazuje spojení zprávou ClientHello, kde:

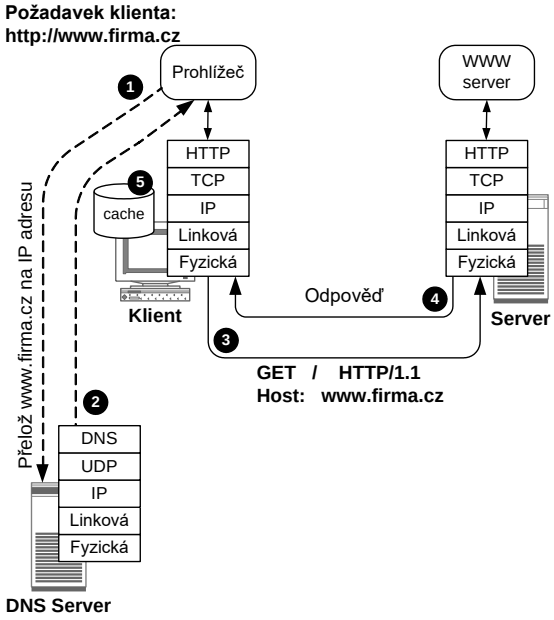
- V záhlaví protokolu RLP uvede verzi 2.
- Použije zprávu ClientHello verze 2, ale v poli verze vyplní verzi 3.1, tj. TLS (resp. 3.0 pakliže neumí TLS, ale podporuje SSL verze 3).
- Server již odpoví protokolem TLS, tj. verzí 3.1 (resp. 3.0 pakliže neumí TLS, ale podporuje SSL verze 3).

15 TLS a HTTP

I když je protokol TLS otevřený k využití libovolným aplikačním protokolem, vznikl jako zabezpečení protokolu HTTP. Nyní si v krátkosti zopakujeme základy protokolu HTTP, abychom si následně mohli objasnit pojem HTTPS.

Základní architekturou komunikace v protokolu HTTP je komunikace klient server. V případě, že se navazuje přímé spojení protokolem TCP mezi klientem a serverem (Obrázek 14.1), pak uživatel zapíše do okna prohlížeče lokátor objektu (URL), jenž si chce prohlížet, a klient nejprve z URL objektu vyřízne jméno serveru, jež přeloží za pomocí DNS na IP-adresu (1 a 2). Poté klient naváže s takto získanou IP-adresou serveru spojení protokolem TCP

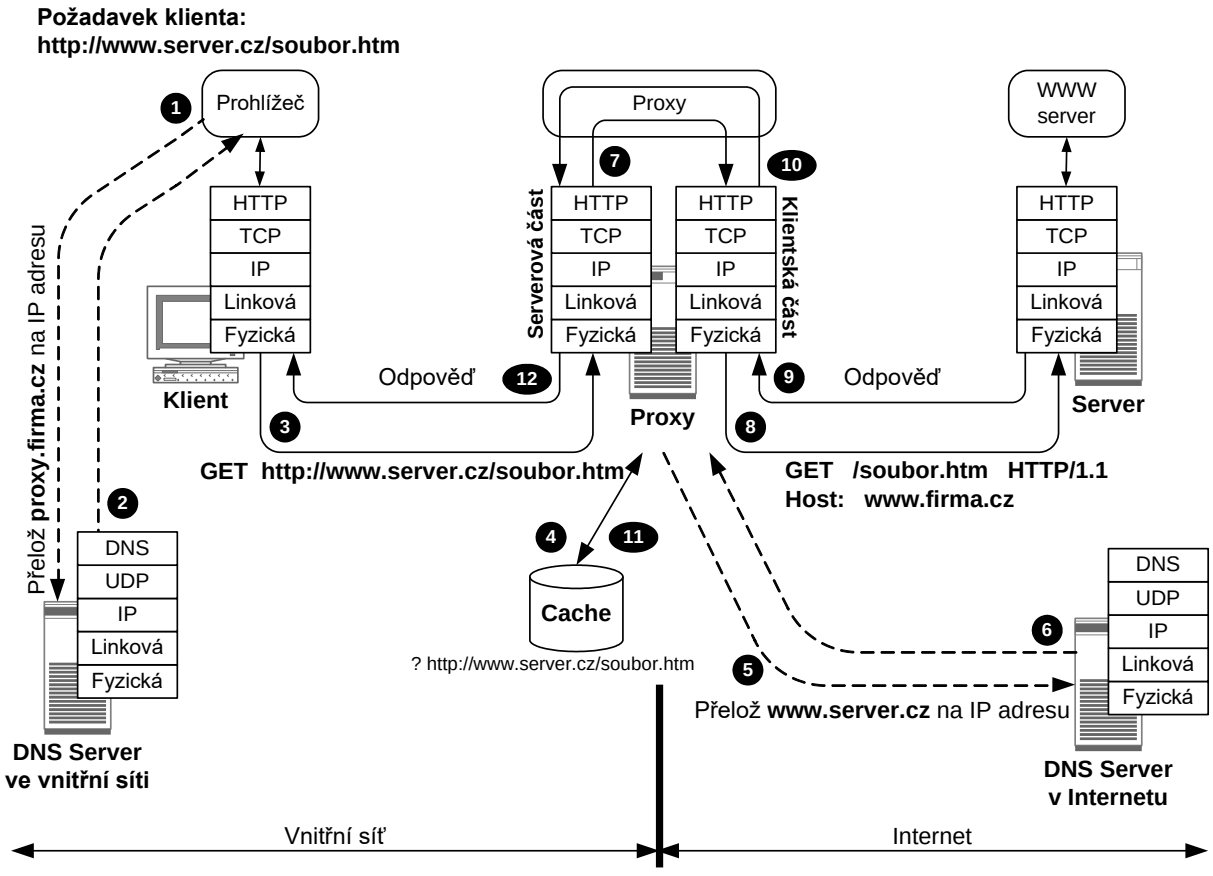
Do takto vytvořeného kanálu vloží HTTP klient dotaz 3 , na který v témž spojení HTTP server odpoví odpovědí 4. Prohlížeč následně zobrazí odpověď uživateli. Získanou odpověď serveru však klient může též uložit i do paměti cache, aby při opakované odpovědi na týž dotaz mohl uživatele uspokojit rychleji.



Obrázek 14.1 Architektura protokolu HTTP

15.1 Proxy

Protokol HTTP zavádí proxy, bránu a tunel. Jedná se o mezilehlé systémy, které mohou ležet na cestě



Obrázek 15.2 Proxy (na obrázeku se již v dotazu klienta nevešla verze protokolu a hlavička Host)

mezi klientem a serverem. Na cestě od klienta k serveru může ležet libovolné množství těchto mezilehlých systémů. Z hlediska protokolu TCP se navazuje TCP spojení vždy mezi dvěma uzly. Tj. TCP spojení mezi klientem a první proxy, mezi první proxy a druhou proxy atd. Při popisu proxy, brány a tunelu se omezím nejprve na situaci, že mezi klientem a serverem je pouze jeden mezilehlý systém. Následně si pak řekneme, že umístěním více mezilehlých systému se vůbec nic nezmění.

Nejčastěji se proto tyto systémy používají tam, kde není možné přímo navázat TCP spojení mezi klientem a serverem. Tj. např. když mezi klientem a serverem leží firewall oddělující intranet od Internetu.

Proxy je systém skládající se ze dvou částí:

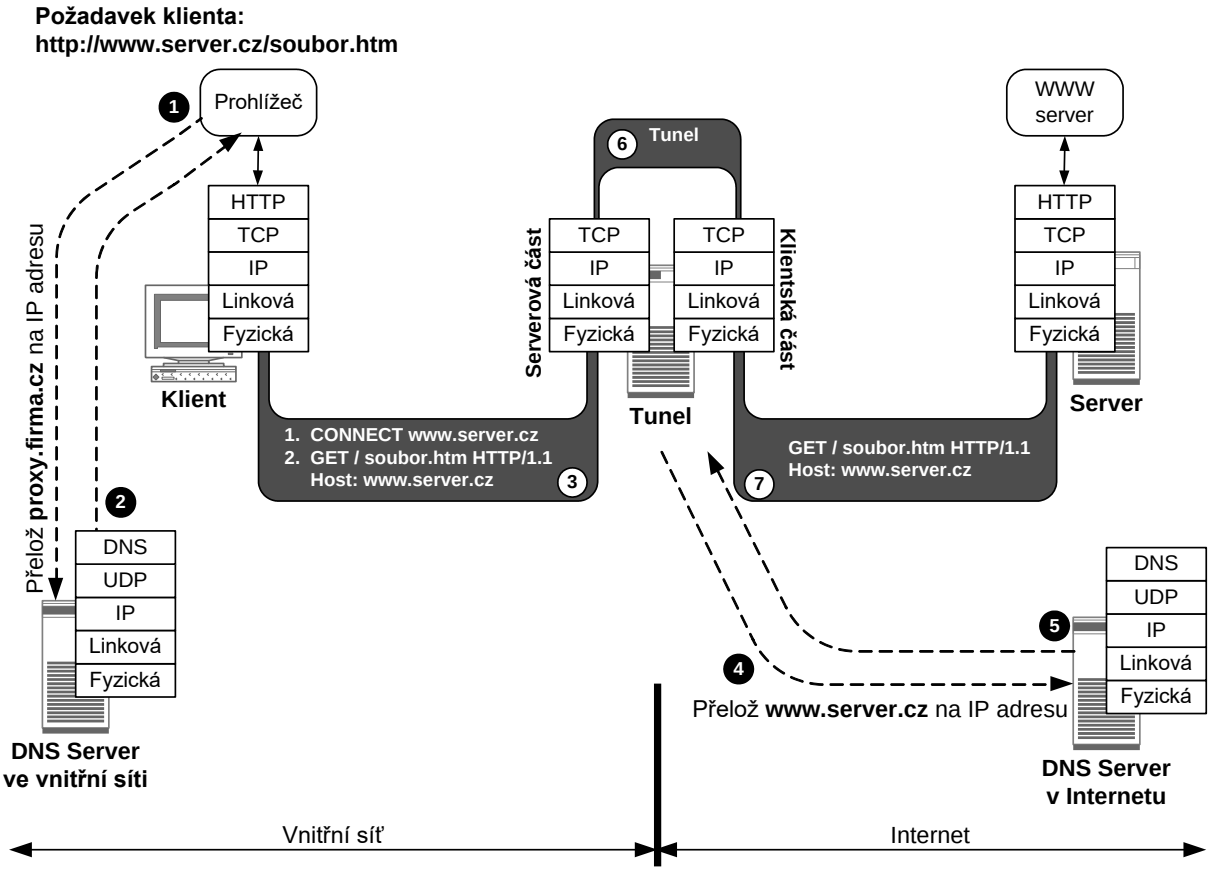
1. Ze serverové části, která přijímá požadavky klienta jakoby je přijímal cílový server. Požadavky však v zápětí předá klientské části.
2. Z klientské části, která převezme požadavky od serverové části, naváže TCP spojení s cílovým serverem a předá jménem klienta požadavky cílovému serveru k vyřízení.

Takto se Proxy jeví uživateli. Avšak uprostřed Proxy mezi serverovou a klientskou částí je ještě skryta vlastní logika Proxy. Proxy totiž rozumí aplikačnímu protokolu (v našem případě protokolu HTTP) a s přijatým požadavkem od klienta může provést několik operací:

1. Může přepsat požadavek (resp. odpověď), tj. změnit data aplikačního protokolu.
2. Odpovědi může ukládat do paměti cache (např. na disk). Pokud proxy obdrží v budoucnu stejný požadavek (např. i od jiného klienta), pak může vrátit tento požadavek rychleji přímo z paměti, aniž by navazovala spojení s cílovým serverem. Vypadá to sice efektivně, ale zásadní otázkou takto uchovávaných odpovědí je jejich aktuálnost. V dnešní době již máme k dispozici velice sofistikované algoritmy pro práci s pamětí cache.
3. Může zjišťovat zdali klient je oprávněn takový požadavek provést.

Na obr. Obrázek 15.2 je schematicky znázorněna činnost proxy. Na počátku uživatel zapíše do okna svého prohlížeče identifikátor objektu (URL) jež chce prohlížet. Např. klient do okna prohlížeče vloží požadavek:

```
http://www.server.cz/soubor.htm
```



Obrázek 15.3 Tunel

Klient bude však vyřizovat tento požadavek skrze proxy. Tj. v konfiguraci svého prohlížeče má uvedeno jméno proxy skrze kterou se bude požadavek vyřizovat.

V prvním kroku klient přeloží jméno proxy na IP-adresu (1 a 2). Jméno cílového serveru totiž nemusí být v intranetu ani přeložitelné. Nyní klient naváže TCP komunikaci se serverovou částí proxy na portu uvedeném v konfiguraci klienta. Do takto vytvořeného TCP spojení vloží klient svůj HTTP požadavek 3:

```
GET http://www.server.cz/soubor.htm
HTTP/1.1
Host: www.server.cz
```

Proxy ve své paměti cache prověří, zdali odpověď na tento požadavek náhodou nemá k dispozici 4.

V případě, že požadavek v paměti cache nebyl nalezen, tak předá požadavek klientské části na vyřízení. Klientská část musí z URI požadavku vyříznout jméno serveru (`www.server.cz`) a přeložit jej v DNS (5 a 6). Jelikož proxy má již přístup do Internetu, je již schopna tento požadavek nechat přeložit v Internetu.

Klientská část proxy nejprve přepíše požadavek na:

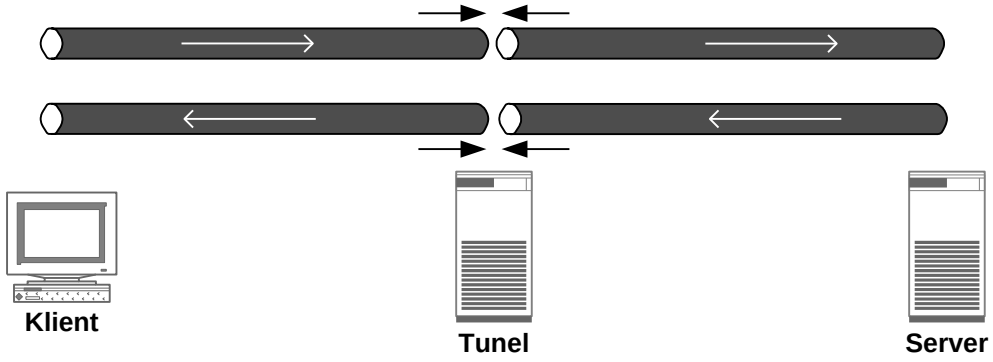
```
GET /soubor HTTP/1.1
Host: www.server.cz
```

Následně klientská část proxy naváže spojení s cílovým serverem protokolem TCP a předá mu požadavek 8 jménem klienta. Server vrátí odpověď 9, kterou obdrží proxy 10. Pokud je odpověď přípustné uložit do paměti cache, pak ji tam uloží 11. Proxy předá odpověď klientovi 12, který ji zobrazí uživateli a případně si ji též uloží do své paměti cache.

15.1.1 Tunel

Tunel nemusí „rozumět“ přenášeným datům. Tunelem lze dokonce přenášet aplikační data zašifrovaná. Toho využívá protokol TLS.

Činnost tunelu je znázorněna na obr. Obrázek 15.3. Klient přeloží jméno tunelu na IP-adresu (1 a 2). Klient naváže s tunelem TCP spojení. Do takto vytvořeného kanálu klient zpravidla vloží pouze příkaz (tj. metodu) CONNECT se jménem a portem cílového serveru 3. Tunel přeloží jméno cílového serveru na IP-adresu (5 a 6) a naváže s cílovým serverem TCP spojení na portu uvedeném v metodě CONNECT.



Obrázek 15.4 Tunel „navaří“ spojení na sebe

Nyní má tunel navázány dvě obousměrná spojení. Každé spojení si představíme jako dvě roury (Obrázek 15.4): jedna roura je pro spojení tam a druhá pro spojení zpět (duplexní spoj).

Tunel neudělá nic jiného, než že roury „navaří na sebe“, tj. tunel, aniž by věděl co přenáší, mechanicky předá do roury na cílový server vše co přijde od klienta. Obdobně vše co přijde ze serveru předá klientovi.

V takovémto spojení pak klient může začít protokolem TLS navazovat zabezpečené spojení. Tj. klient začne úvodní dialog protokolu HP zprávou `ClientHello…` .

Je vcelku pochopitelné, že tunel nevidí do přenášených dat, takže nemůže kontrolovat, co klient ze serveru stahuje za data. Zdali se např. nejedná o Java applety či ActiveX komponenty.

Po ukončení spojení se celý tunel rozpadá.

.

15.2 HTTPS

Pod názvem HTTPS se míní protokol HTTP, který je zabezpečen pomocí TLS (resp. SSL), tj. HTTPS je označení pro „*HTTP over TLS*“. HTTPS je tak synonymu pro „bezpečný web“.

Klient se odkazuje na HTTPS server rovněž pomocí URI, kde je místo modelu http je uveden model https. URI může obsahovat DNS-jméno serveru nebo IP adresu serveru. Klient kontroluje shodu

DNS-jména serveru s údaji v rozšíření Alternativní jméno předmětu v certifikátu serveru. V případě DNS-jména se jméno serveru musí shodovat s alespoň jedním jménem uvedeným v položce `dNSName`. Využití IP adresy v URI je sice výjimečné, v takovém případě se kontroluje shoda IP adresy serveru s položkou `iPAddress` rozšíření Alternativní jméno předmětu v certifikátu serveru.

V jednom certifikátu t může být více alternativních jmen předmětu či může být použit znak hromadného výběru - hvězdička (*). V takovém případě stačí, aby se shodovalo alespoň jedno jméno uvedené v certifikátu s DNS jménem webového serveru. RFC-2818 vyzývá CA, aby upustily od používání DNS jména v předmětu certifikátu v relativním jedinečném jméně *Common Name*.

V případě, že klient nenalezne shodu s údaji v certifikátu serveru, je povinen na tuto skutečnost upozornit uživatele nebo ukončit spojení. V praxi je běžnější upozornění uživatele, kterému se zobrazí pro něj často nepochopitelné hlášení byť obsahující několik vykřičníků. Uživatele v podstatě tato hlášení obtěžují a tak s vysokou pravděpodobností toto hlášení budou ignorovat a budou pokračovat rovněž s vysokou pravděpodobností na podvrženém serveru.

15.3 Protocol upgrade

Předpokládejme, že klient navázal spojení s webovým serverem protokolem TCP např. na portu 80/tcp. Změna protokolu na bezpečný protokol se provádí vždy z iniciativy klienta. Dokonce i v případě, že server z bezpečnostních důvodů vyžaduje dále komunikaci skrze vrstvu TLS, pak pouze klientovi vrátí zprávu:

```
HTTP/1.1 426 Upgrade Required
Upgrade: TLS/1.0, HTTP/1.1
Connection: Upgrade
```

A klient je povinen začít dialog o přechodu na použití vrstvy TLS. Např.:

```
OPTIONS * HTTP/1.1
Host: www.firma.cz
Upgrade: TLS/1.0
Connection: Upgrade
```

V případě, že klient sám od počátku chce přejít na použití vrstvy TLS, pak začne úvodní dialog přímo:

```
GET / HTTP/1.1
Host: www.firma.cz
Upgrade: TLS/1.0
Connection: Upgrade
```

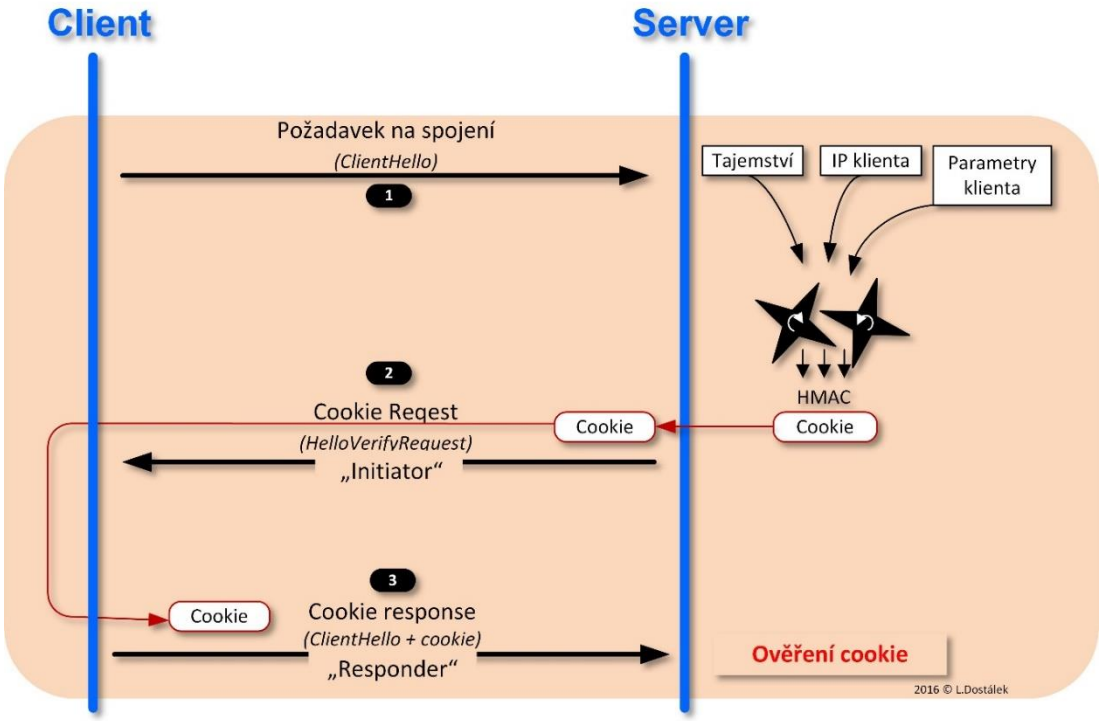
Server odpoví, že souhlasí s přechodem na vrstvu TLS, např.:

```
HTTP/1.1 101 Switching Protocols
Upgrade: TLS/1.0, HTTP/1.1
Connection: Upgrade
```

Poté následují zprávy `ClientHello`, `ServerHello` a vůbec zřízení TLS (resp. SSL) relace. Navázanou relací pak server pokračuje ve své odpovědi.

16 Photuris

Photuris je latinský název rodu severoamerických světlušek, jejichž dospělé samičky napodobují světelné signály jiných druhů světlušek, aby přilákaly jejich samečky, a pak je sežraly.



obr. 16.1 Mechanismus cookie

Protokol Photuris byl vtvořen pro dnes již zapomenutá zařízení. Zůstal z něj nápad na způsob obrny proti DoS (DDoS) útokům pomocí cookie, které mj. využívají protokoly DTLS a SCTP.

Klasický útok na protokol TCP je *SYN flood attack*. Při tomto útoku útočník útočí na server TCP pakety s nastaveným příznakem SYN. Server okamžitě alokuje datové struktury pro navázání komunikace. Jestliže je paketů s nastaveným příznakem SYN velké množství, tak server může zkolabovat vyčerpáním alokovaných zdrojů.

V případě datagramových služeb je možnost útoku podstatně větší než v případě protokolu TCP. Bylo by tedy dobré útočníka nějak potrápit ještě před tím, než se alokují patřičné zdroje.

Protokol Photuris přišel s řešením, že server vygeneruje cookie, které zašle klientovi, který musí svůj požadavek znovu zopakovat, ale se zkopírovanou cookie. Teprve pak je požadavek např. na zřízení relace akceptován.

To je docela dobrá ochrana proti DoS útokům prováděným z neexistujících nebo falešných IP adres. Z existujíc ích IP adres je možné správně odpovědět, ale vyžaduje to, aby útočník rozuměl danému protokolu.

Systém byl vylepšen tak, že se cookie negeneruje prostým generátorem náhodných čísel, ale server si nejprve vygeneruje tajemství. Vytvoří se struktura, která obsahuje IP adresu klienta a další parametry spojení s klientem. Z této struktury se spočte HMAC.

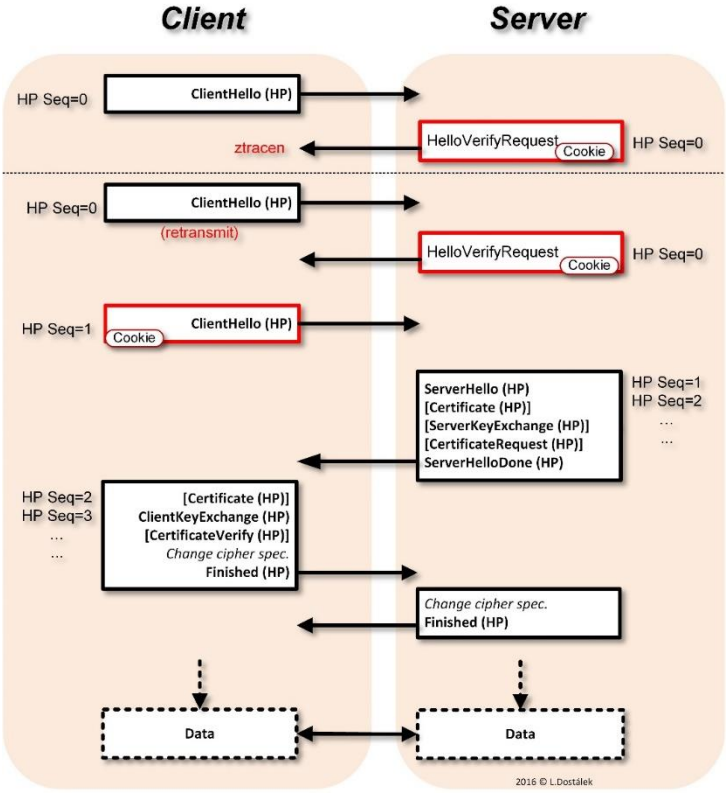
Protokol DTLS používá tzv. bez stavové cookie, která obsahuje HMAC. Protokol SCTP používá tzv. stavové cookie, kde cookie tvoří celá datová struktura včetně HMAC. DTLS po odeslání stavové cookie dealokuje všechny datové struktury, které vytvořil na příchozí požadavek a čeká, jestli dostane správnou odpověď.

Photuris

Světluška větší (naše)

17 DTLS

Protokol TLS (*Transport Layer Security*) je velice dobře prověřený protokol, jsou známy útoky na něj atd. Má však jednu nevýhodu – zabezpečuje komunikaci přes TCP protokol. Tj. není určen pro zabezpečení datagramové komunikace. Tento problém řeší protokol DTLS (*Datagram Transport Layer Security*), který modifikuje protokol TLS pro datagramové služby.



Obrázek 17.1 Dialog protokolu HP modifikovaný pro DTLS

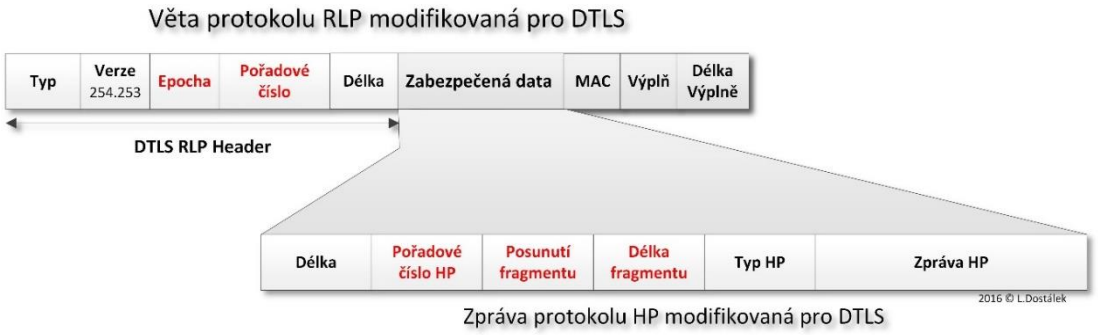
Cílem protokolu DTLS je minimalizovat změny protokolu TLS tak, aby byl uzpůsoben zabezpečení datagramových toků. Vše, co je možné ponechat z TLS, bylo ponecháno.

Hlavní rozdíly oproti TLS:

- Není možné používat proudové šifry (např. RC-4), protože při ztrátě paketu by nebylo možné následující pakety dešifrovat.
- TLS nepotřebuje číslovat pakety, protože TCP zajišťuje jejich plynulé předávání. V terminologii DTLS se říká, že RLP pakety jsou v TLS implicitně číslovány. Jelikož to v případě datagramového toku neplatí (některé pakety se mohou ztratit), tak je třeba datagramy v toku explicitně číslovat. Čísluje se:
  - Jednotlivé věty protokolu RLP (Obr8zek 17.2) se číslují v položce Pořadové číslo. Položka Epocha se mění vždy změně šifrování. Je to z proto, že datagram obsahující změnu šifrování se může ztratit.
  - Jednotlivé zprávy protokolu HP se rovněž explicitně číslují. To je také z důvodu, že datagram nesoucí zprávu protokolu HP se může ztratit. Při odeslání zprávy protokolu HP se aktivuje časový čítač, který měří čas došlé odpovědi, pokud odpověď včas nepřijde, tak se zpráva protokolu HP zopakuje. Zopakuje se se původním číslem zprávy protokolu HP. Na obr. 17.1 se na počátku zpráva ClientHello včas nedočkala odpovědi, tak byla zopakována, ale se stejným číslem HP zprávy (HP seq).
- Zprávy protokolu HP mohou být fragmentovány, proto do záhlaví zprávy protokolu HP byly

přidány položky Posunutí fragmentu a Délka fragmentu.

- Jako obrana proti DoS (DDoS) útokům byla zvolena bez stavová cookie. Přibyla tak nová zpráva HelloVerifyRequest, jejímž cílem je zaslat cookie. Klient začíná dialog zprávou Clienthello (Obrázek 17.1), která neobsahuje cookie, server odpoví zprávou HelloVerifyRequest s cookie. Klient přidá do původní zprávy ClientHello cookie a dialog protokolu HP běží již dálo, jako ho známe z TLS.



obr. 17.2 Věta protokolu RLP a zpráva HP, obě modifikovány pro DTLS
(červeně jsou vyznačeny odlišnosti od TLS)

18 Kerberos

Kerberos vznikl v dávných dobách, kdy internet byl doménou akademické sféry. V těch dávných dobách se uživatelé k serverům přihlašovali hesly, která se po síti přenášela v textovém tvaru. Kerberos měl řešit jednak zabezpečení přenosu hesla sítí a jednak vytvoření společné autentizace pro více aplikací – tj. to co dnes známe pod označením Single Sign On (SSO). Kerberos si žil svým životem vcelku bez velkého rozšíření, kterému jednak bránila exportní omezení USA, a jednak nutnost úpravy aplikací tak, aby podporovaly autentizaci pomocí Kerbera – tzv. kerberizaci aplikací. Zlom přinesly Windows 2000, které vedle nativní autentizace NTLM implementovaly autentizaci Kerberem, tj. všechny aplikace Microsoftu byly najednou kerberizovány.

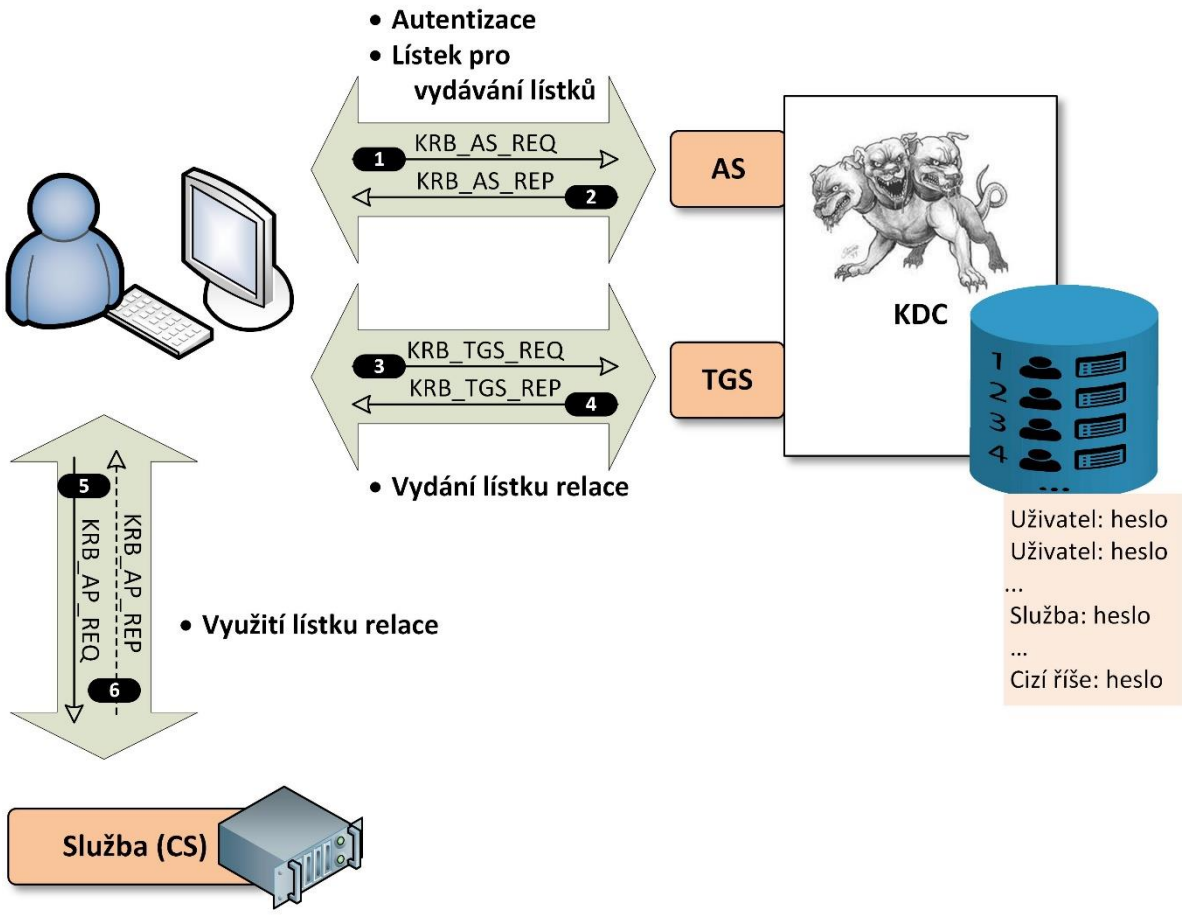
18.1 Princip

Princip je založen na myšlence, že místo hesla budeme sítí přenášet nějakou zprávu tímto heslem šifrovanou. Kerberos zavádí zajímavou terminologii:

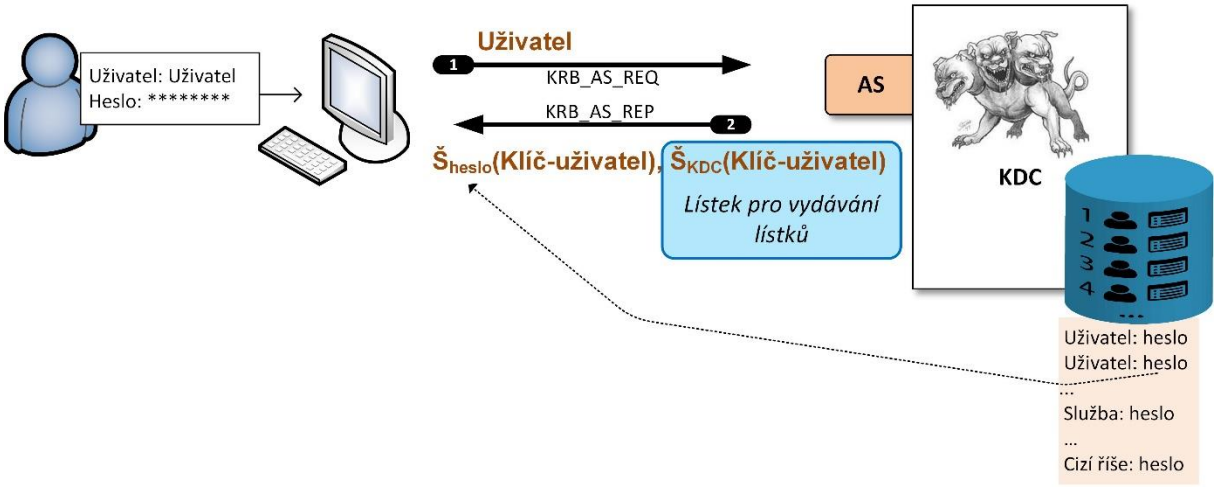
- Pro uživatele a službu používá termín principál.
- Množina principálů, pro které Kerberos zajišťuje autentizaci nazývá říší *(realm)*.

V případě Windows je říše shodná s doménou Windows (ta zase může být shodná s doménou DNS). Windows používají termíny *User Principal Name* (UPN) pro principály uživatele a *Service Principal Name* (SPN) pro principály služby. Pro UPN se používá notace, kterou známe z e-mailových adres, tj. uživatel@doména. U SPN se používá stejná notace, ale musíme si uvědomit, že musíme najít označení pro službu běžící na serveru, tj. část musí před @ obsahovat jak název služby, tak i název serveru. Příklad SPN: `http/www.firma.cz@firma.eu` – služba `http` na serveru `www.firma.cz`, který je vedený v říši `firma.eu`).

Základem Kerbera je nezávislá třetí strana Server pro vydávání klíčů - *Key Distribution Server* (KDC), který se skládá z:



Obrázek 18.1 Základní komunikace protokolu Kerberos



Obrázek 18.2 Autentizace uživatele

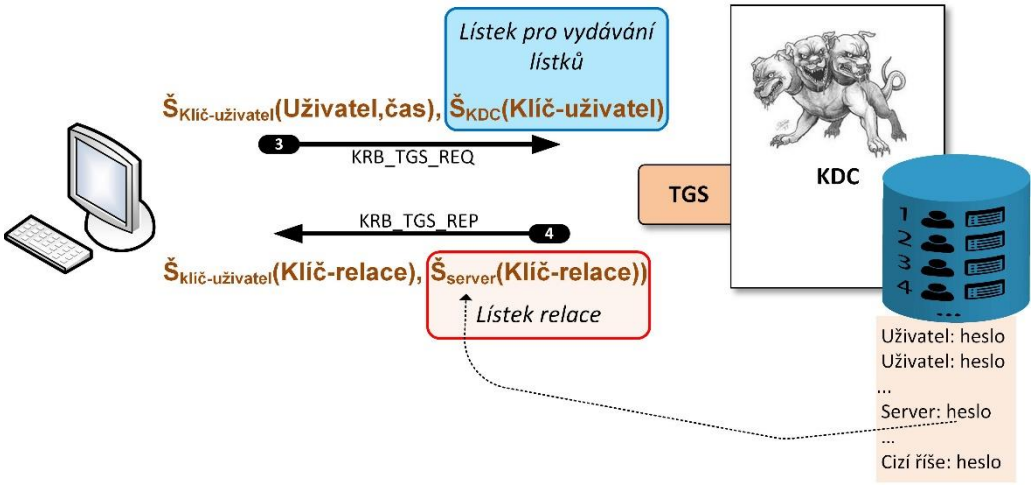
- Databáze principálů, jejich autentizačních metod, jejich hesel a dalších údajů o nich.
- Autentizačního serveru (AS), který provádí vlastní autentizaci uživatelů. Výsledkem autentizace užávatele je vydání tzv. lístku pro vydávání lístků.
- Služby pro vydávání lístků - *Ticket Granting Service* (TGS) - pokud uživatel chce následně přistoupit na konkrétní službu (např. MS Exchange), pak si u komponenty *Ticket Granting Service* (TGS) nechá na základě lístku pro vydávání lístků vydat lístek relace pro konkrétní službu. S tímto lístkem se pak autentizuje vůči příslušné požadované službě.

Důležité je, že uživatel (člověk) zadává jenom jednou své autentizační údaje (např. jméno a heslo) a celý mechanismus vydávání a uplatňování lístků už za něj dělá software. Prakticky: uživatel (člověk) ráno přijde do práce, autentizuje se, a jeho počítač už na pozadí implementovaným mechanismem vydávání/uplatňování lístků sám uživatele přihlašuje do všech aplikací, kam uživatel (člověk) celý den potřebuje přistoupit.

18.2 Autentizace

Uživatel se přihlašuje pomocí jména a hesla. Software za uživatele vytvoří zprávu, která mj. obsahuje jméno uživatele (*User Principal Name*), ale neobsahuje zadané heslo – to si software uloží do keše. Autentizační server (AS) příjme tuto zprávu, v KDC vyhledá uvedeného uživatele a následně získá jeho heslo. Uživateli vygeneruje `Klíč-uživatel`, který šifruje:

- Jednak heslem uživatele, tak aby si jej uživatel mohl dešifrovat a mít k němu přístup.



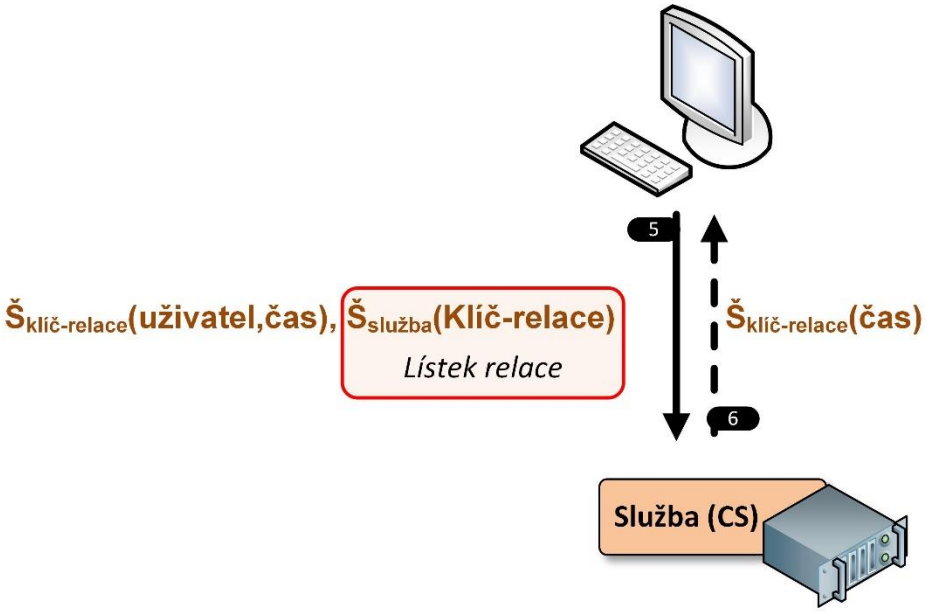
Obrázek 18.3 Vydání lístku relace

- Svým heslem (heslem KDC), takto zašifrovaný `Klíč-uživatel` tvoří tzv. lístek pro vydávání lístků.

Autentizační server následně uživateli odešle

Tímto klíčem dešifruje dvojici uživatel a čas, pokud je vše v pořádku, pak je uživatel autentizován.

Služba TGS autentizovanému uživateli vrací:



Obrázek 18.4 Uplatnění lístku relace

strukturu, která obsahuje jednak heslem uživatele šifrovaný `Klíč-uživatel` a jednak lístek pro vydávání lístků. Autentizace uživatele se provede na základě úvahy: jedině uživatel, který zná správné heslo je schopen dešifrovat „heslem uživatele šifrovaný `Klíč-uživatel`" a získat tak `Klíč-uživatel` pomocí kterého se software uživatele jménem uživatele bude autentizovat v následné komunikaci.

V okamžiku, kdy software klienta získá `Klíč-uživatel`, tak může z své keše vymazat heslo.

Lístek pro vydávání lístků (včetně klíče `Klíč-uživatel`) má zpravidla platnost několika hodin (viz odstavec Časy).

Lístek relace

Nyní uživatel chce přistoupit ke konkrétní službě. Tj. od TGS chce vydat lístek relace pro přístup na konkrétní službu. Software uživatele vytvoří žádost o vydání lístku relace obsahující:

- Klíčem `Klíč-uživatel` šifrovanou dvojici jméno uživatele a čas.
- Lístek pro vydávání lístků.

TGS nejprve heslem KDC dešifruje lístek pro vydávání lístků, čímž získá `Klíč-uživatel`.

- `Klíč-relace` pro autentizaci uživatele k příslušné službě, jehož přenos je chráněn tím, že je šifrován pomocí `Klíč-uživatel`.
- Lístek relace, kterým se uživatel bude autorizovat vůči požadované službě. Lístek relace obsahuje klíč relace šifrovaný heslem služby.

Poznámka: `Klíč-uživatel` je též klíčem relace – relace mezi KDC a uživatelovým klientském software (klientem). V literatuře se proto zpravidla i pro `Klíč-uživatel` používá termín `Klíč-relace`. Používám dva termíny, protože mi pak výklad připadá pochopitelnější.

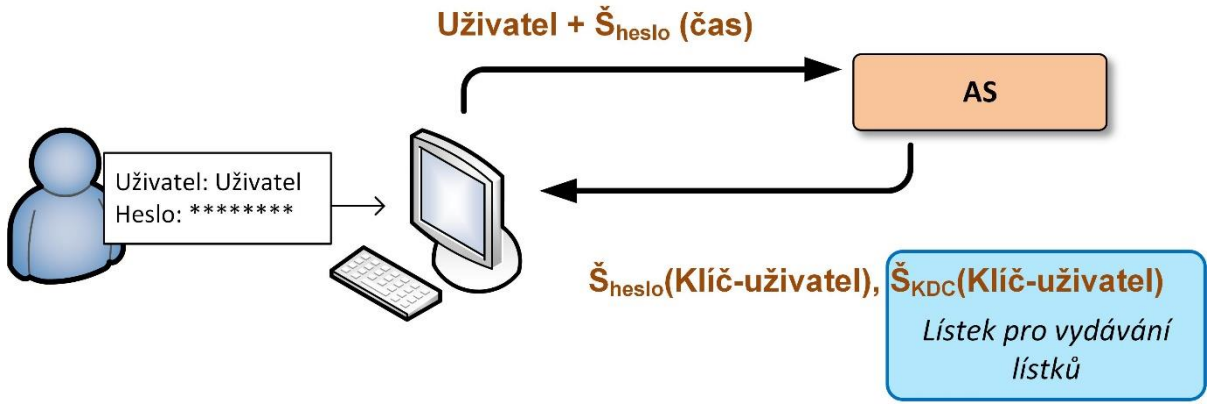
Přihlášení uživatele k požadované službě

Uživatel se autentizuje vůči službě tím, že pošle:

- Klíčem relace `Klíč-relace` šifrovanou dvojici jméno uživatele a čas.
- Lístek relace, který obsahuje klíč relace `Klíč-relace` šifrovaný heslem služby.

Služba si dešifruje lístek relace, získá tak dešifrovací k dešifrování dvojice uživatel a čas. Pokud je vše v pořádku, pak je uživatel autentizován. Na závěr se ještě se může služba autenti-

zovat uživateli tak, že klíčem relace šifruje datovou strukturu obsahující čas. V takovém případě dojde k oboustranné autentizaci klienta i služby.



Obrázek 18.5 Před-autentizace

18.3 Před-Autentizace

Zatím jsem předpokládal, že autentizace probíhá tak, že uživatel odešle na Autentizační server své jméno (*User Principal Name*), Autentizační server vygeneruje `Klíč-uživatel`, který šifruje heslem uživatele a zašle uživateli zpět. Jedině uživatel znalý svého hesla si může tuto zprávu dešifrovat a získat `Klíč-uživatel`. Tento způsob autentizace má drobnou vadu - útočník může posílat na Autentizační server jména libovolných uživatelů a Aplikační server mu bude vracet odpovědi, které může útočník dále zkoumat. Např. pokud uživatel použije slabé heslo, tak se jej může pokusit zlomit.

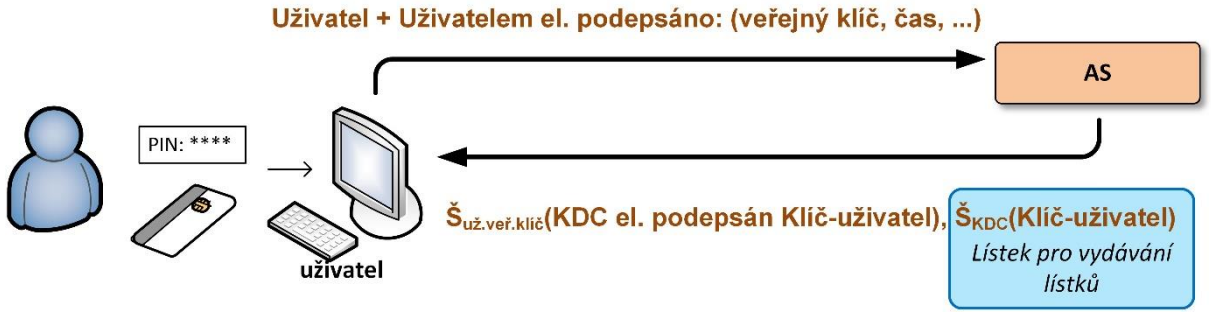
Je tedy třeba zamezit tomu, aby libovolný neautentizovaný uživatel mohl na Autentizační server posílat požadavky na autentizaci a Autentizovaný server mu automaticky odpovídal. Řešením je tzv. před-autentizace. Ta je zpravidla založena na tom, že klient na Autentizační server, kromě svého jména, posílá i svým heslem šifrovanou datovou strukturu, která obsahuje čas. Autentizační server se pak tuto strukturu snaží dešifrovat heslem uživatele a následně odpovídá lístkem pro vydávání lístku jen takovému uživateli, který se před-autentizoval, tj. jehož datovou strukturu obsahující čas Autentizační server úspěšně dešifroval. Nutno ale dodat, že před-autentizačních mechanismů existuje celá řada.

18.4 Další způsoby autentizace

Díky tomu, že Autentizační server (AS) je samostatnou komponentou, tak je jej možné implementovat pro nejrůznější způsoby autentizace. Pro autentizaci je možné použít např. nejrůznější hardwarové autentizační kalkulátory. Dodavatelé takových zařízení pak zpravidla dodají jednak hardwarový prostředek včetně software pro klienta, ale také vlastní implementaci AS včetně kryptografického materiálu, který je sdílen mezi hardwarovým autentizačním prostředkem a AS.

Také jsou podporovány autentizace pro nejrůznější algoritmy jednorázových hesel a též autentizace čipovou kartou.

Ať už je autentizace založena na jakémkoliv principu, tak výsledek musí být vždy klíč `Klíč-uživatel` a lístek pro vydávání lístků.



Obrázek 18.6 Autentizace čipovou kartou

18.5 Autentizace čipovou kartou

Autentizace čipovou kartou je spojena s asymetrickou kryptografií. Na čipové kartě je soukromý klíč a též veřejný klíč obsažený v příslušném certifikátu. Asymetrická kryptografie je výhradně spojena s autentizací uživatele. Jakmile je vydán klíč `Klíč-uživatel` a lístek pro vydávání lístků, tak vše dále již běží, jak bylo dříve popsáno, tj. na bázi symetrické kryptografie.

Windows využívají autentizaci čipovou kartou i pro přihlášení uživatele do systému. Uživatel vloží kartu do čtečky a zadá přístupový PIN k soukromému klíči na kartě, tj. vše běží bez hesel. Otázkou je, kde se vezme jméno uživatele (tj. *User Principal Name* - UPN), když uživatel zadává jen PIN. To je uloženo v příslušném certifikátu veřejného klíče v Alternativním jméně předmětu *(Subject Alternative Name)* v položce Jiné jméno *(Other Name)*.

$Š_{heslo}$

V předchozím textu jsem napsal „šifrováno heslem“. Jedná se jen o silné zjednodušení pro snadné pochopení. Je zjevné, že heslem nelze šifrovat, protože na šifrovací klíč jsou kladeny požadavky konkrétními šifrovacími algoritmy. Slovním spojením „šifrováno heslem“ jsem mínil to, že příslušný tajný klíč bude odvozen (derivován) z hesla jednocestnou funkcí tak, aby vyhovoval příslušnému algoritmu, aby odvozování klíče z hesla nebylo reverzibilní a pokud možno byla zachována entropie hesla. Operace, kterou se převádí heslo na tajný klíč se označuje jako *„string-to-key“* operace (viz např. legendární standard PKCS#5).

KDC používá pro zabezpečení komunikace s konkrétním principálem tzv. typ šifrování *(Encryption Type)*. Jedná se vždy o sadu kryptografických protokolů, která obsahuje nejenom šifrovací algoritmus, ale i hešovací algoritmus, případně hešovací algoritmus v kombinaci s tajným klíčem (např. HMAC), které se používají pro *„string-to-key“* operace. Přehled typů šifrování je uveden v následující tabulce:

Encryption Type	Typ šifrování	Zavedeno standardem
1	des-cbc-crc	[RFC3961]
2	des-cbc-md4	[RFC3961]
3	des-cbc-md5	[RFC3961]
4	Reserved	[RFC3961]
5	des3-cbc-md5	
7	des3-cbc-sha1	
9	dsaWithSHA1-CmsOID	[RFC4556]
10	md5WithRSAEncryption-CmsOID	[RFC4556]
11	sha1WithRSAEncryption-CmsOID	[RFC4556]
12	rc2CBC-EnvOID	[RFC4556]
13	rsaEncryption-EnvOID	[RFC4556]
14	rsaES-OAEP-ENV-OID	[RFC4556]
15	des-ede3-cbc-Env-OID	[RFC4556]
16	des3-cbc-sha1-kd	[RFC3961]
17	aes128-cts-hmac-sha1-96	[RFC3962]
18	aes256-cts-hmac-sha1-96	[RFC3962]
19	aes128-cts-hmac-sha256-128	[RFC8009]
20	aes256-cts-hmac-sha384-192	[RFC8009]
23	rc4-hmac	[RFC4757]
24	rc4-hmac-exp	[RFC4757]
25	camellia128-cts-cmac	[RFC6803]
26	camellia256-cts-cmac	[RFC6803]

18.6 Kontrolní součet

Kontrolní součet zajišťuje integritu zprávy. Kerberos dnes požaduje, aby dvě různé zprávy o stejném obsahu měly různý kontrolní součet. Chce tím zabránit, aby útočník z rovnosti kontrolních součtů poznal, že se jedná o týž obsah (bez znalosti tajného klíče). Z tohoto důvodu se též používají kromě hešovacích algoritmů i hešiovací algoritmy v kombinaci s tajným klíčem (např. HMAC). Používané typy kontrolních součtů jsou uvedeny v následující tabulce:

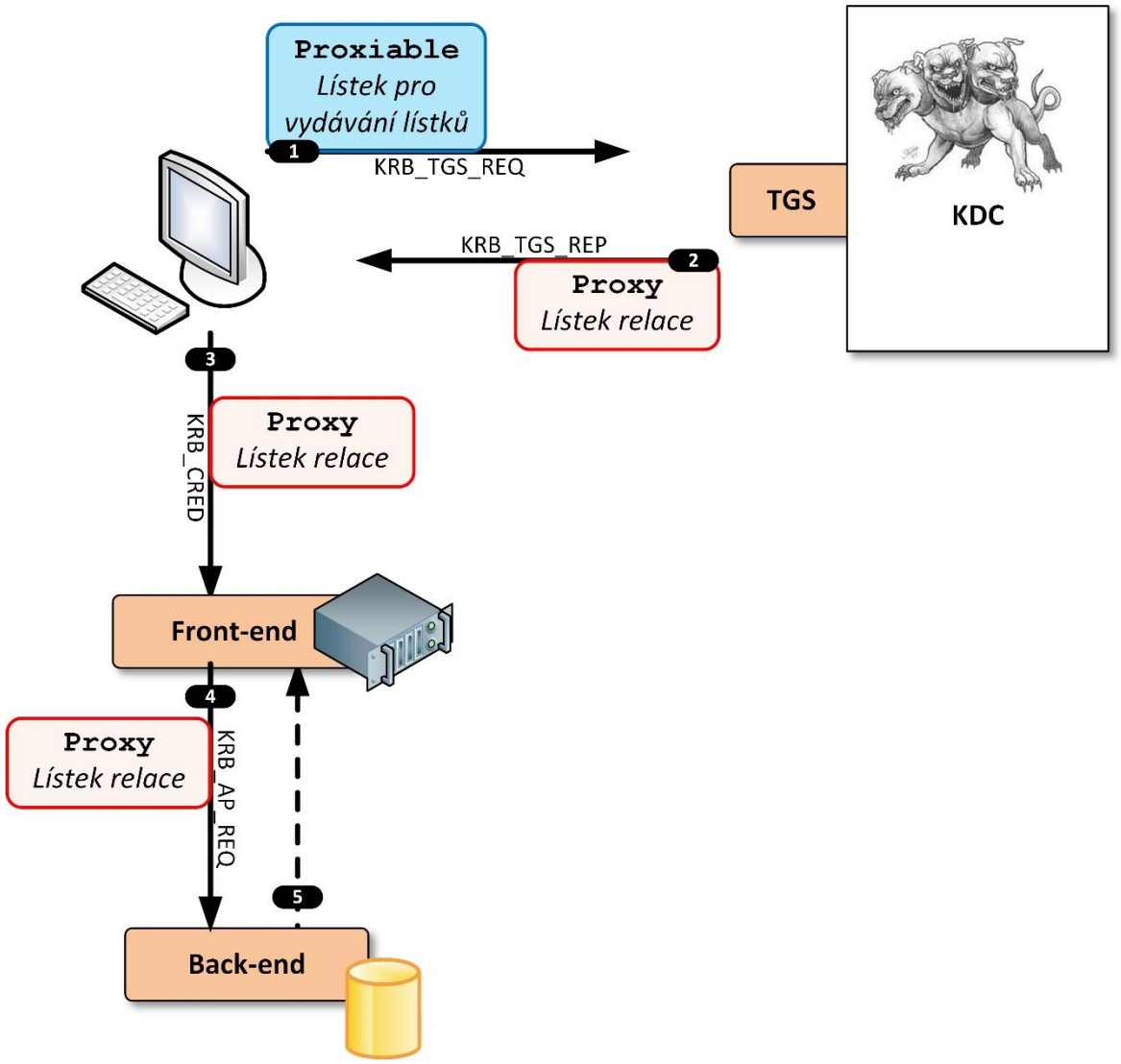
Type	**Typ algoritmu kontrolního součtu**	**Velikost kontrolního součtu**	**Zavedeno standardem**
1	CRC32	4	[RFC3961]
2	rsa-md4	16	[RFC3961]
3	rsa-md4-des	24	[RFC3961]
4	des-mac	16	[RFC3961]
5	des-mac-k	8	[RFC3961]
6	rsa-md4-des-k	16	[RFC3961]
7	rsa-md5	16	[RFC3961]
8	rsa-md5-des	24	[RFC3961]
9	rsa-md5-des3	24	
10	sha1 (unkeyed)	20	
11	Unassigned		
12	hmac-sha1-des3-kd	20	[RFC3961]
13	hmac-sha1-des3	20	
14	sha1 (unkeyed)	20	
15	hmac-sha1-96-aes128	20	[RFC3962]
16	hmac-sha1-96-aes256	20	[RFC3962]
17	cmac-camellia128	16	[RFC6803]
18	cmac-camellia256	16	[RFC6803]
19	hmac-sha256-128-aes128	16	[RFC8009]
20	hmac-sha384-192-aes256	24	[RFC8009]

18.7 Dlouhodobé a krátkodobé klíče

Termínem dlouhodobý klíč rozumíme klíč odvozený od hesla, protože pro konkrétní typ šifrování se od téhož hesla vždy odvodí stejný tajný klíč. Heslo je zpravidla platné delší dobu, tak odtud plyne termín „dlouhodobý“. Krátkodobý klíč je klíč relace (`Klíč-uživatel` je též klíč relace - relace KDC s uživatelem), který je platný po dobu platnosti lístku. Při obnovení lístku se vždy generuje nový krátkodobý klíč. (Pikantní na této terminologii je skutečnost, že tato terminologie se používá i v případě autentizace uživatele pomocí jednorázových hesel, kdy jednorázové heslo je to dlouhodobé.)

Pokud software kešuje heslo, pak zpravidla nekešuje samotné heslo, ale dlouhodobý klíč. Pokud hovoříme o klíči v kontextu Kerbera, pak tím vždy míníme dvojici typ šifrování + klíč. Tj. software kešuje dvojici typ šifrování + dlouhodobý klíč (pokud se vůbec kešuje).

Jelikož si uživatel může např. změnit heslo, tak u šifrovaných struktur se uvádí též verze použitého klíče. Změna verze klíče pak signalizuje, že klíč byl odvozen (derivován) již z jiného hesla. Tj. pokud by software kešoval heslo, pak by kešoval jeho verzi + výše uvedenou dvojici typ šifrování + dlouhodobý klíč.



Obrázek 18.7 Proxy

18.8 Proxy

Proxy se používá k řešení dosud v Čechách rozšířeného nešvaru, kdy Front-end server přistupuje na Back-end (resp. do databáze) pomocí „technického uživatele“, tj. na back-end (resp. do databáze) se nepřenese informace: jménem jakého uživatele byla operace uskutečněna.

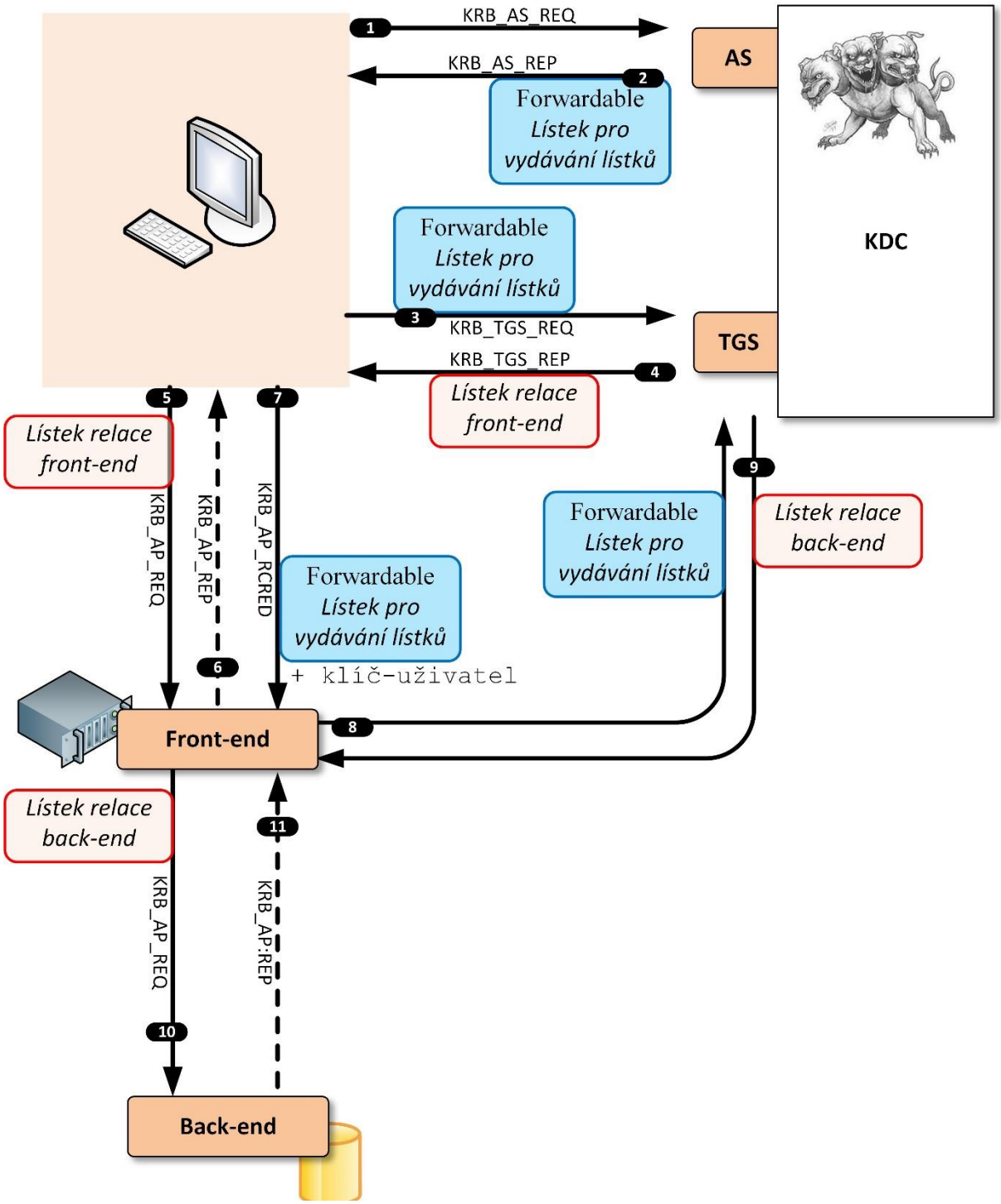
PROXY lístky se používají právě v případě, když klient chce vydat lístek, který následně předá front-end serveru, aby jeho jménem přistoupl na back-end server. Aby se ztížilo využívání kradených lístků, tak jsou obecně lístky často vázány na síťovou adresu. PROXY lístek je tedy lístek relace, který bude požit z jiné síťové adresy než je adresa klienta, proto back-end může při použití PROXY lístků vyžadovat dodatečnou autentizaci fornt-end serveru.

Proxy vyžaduje, aby klient znal jméno služby (SPN) na back-end serveru, protože jej musí zadat při vystavování lístku relace.

18.9 Forwarding

Forwarding je jistou obdobou Proxy, ale klient nemusí znát SPN cílového serveru, protože klient deleguje pravomoc vydávat lístky jeho jménem fornt-end serveru.

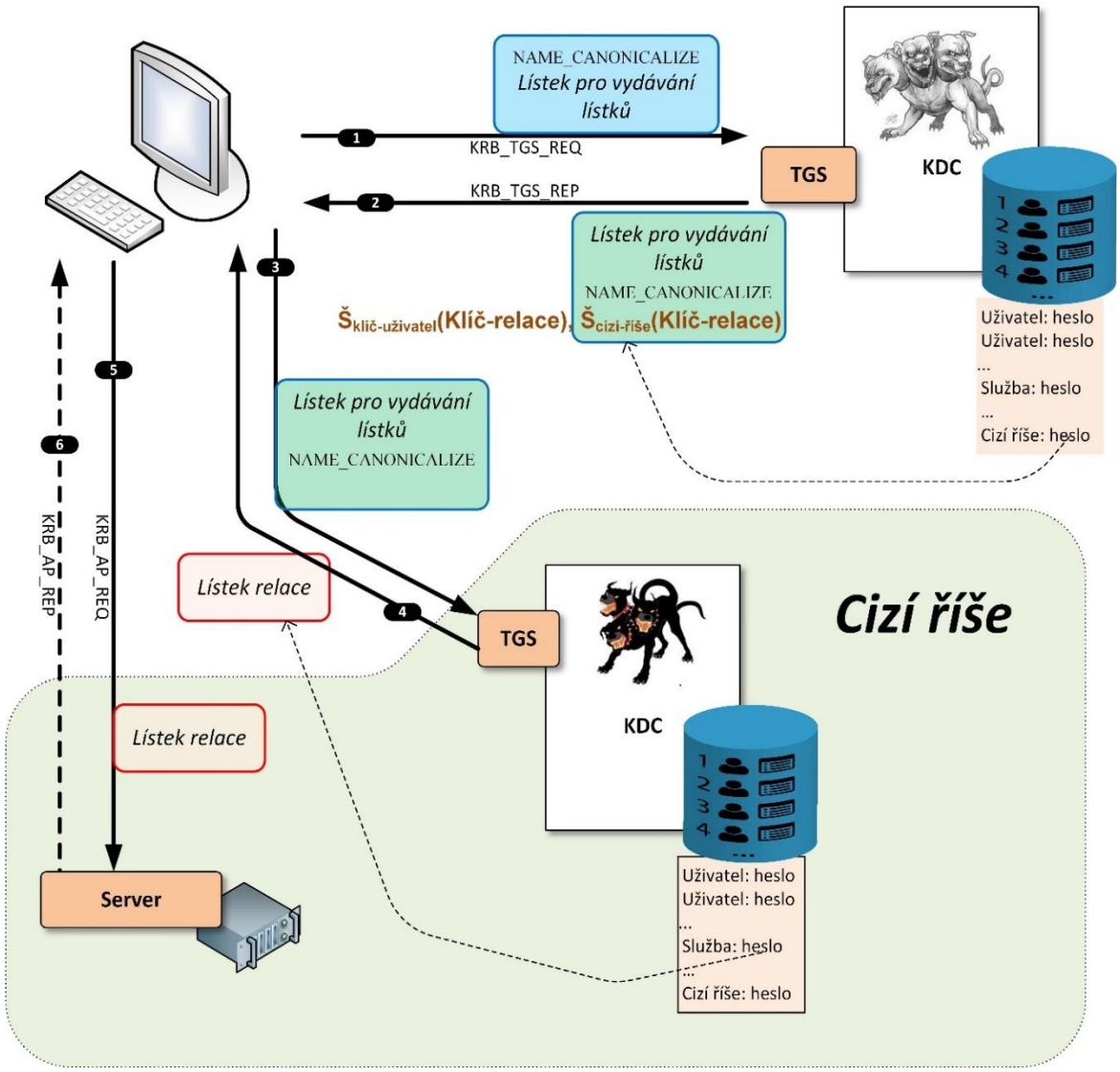
V tomto případě musí mít klient od Autentizačního serveru vydán lístek pro vydávání lístků (TGT) s příznakem FORWARDABLE. Takový TGT může klient následně předat foront-end serveru, aby jménem klienta požádal TGS o vydání lístku pro back-end server. Takto obdržený lístek pro back-end server získá příznak FORWARDED.



Obrázek 18.8 Forwarding

Postup je následující:

1. Klient se autentizuje, požaduje vydání lístku pro vydávání lístků (TGT) pro svou síťovou adresu a pro adresu front-end serveru s příznakem FORWARDABLE.
2. KDC vydá požadovaný lístek.
3. Klient žádá pomocí vydaného TGT lístek relace pro přístup na fornt-end server.
4. KDC vydá požadovaný lístek.



Obrázek 18.9 Důvěra mezi říšemi

5. Klient se autentizuje pomocí vydaného lístku na fron-end a požaduje oboustrannou autentizaci.
6. Front-end se autentizuje vůči klientovi. Výsledkem je, že klient i fornt-end jsou vzájemně autentizováni.
7. Klient odešle na front-end svůj TGT a `Klíč-uživatel`. Tj. deleguje svou identitu na front-end.
8. Front-end využije získaný klientův TGT, aby si na KDC nechal vydat lístek relace pro back-end.
9. KDC vydá lístek front-end serveru s příznakem FORWARDED.
10. Front-end využije vydaný lístek relace pro přihlášení na back-end.
11. Pokud je vyžadována oboustranná autentizace, tak back-end se též autentizuje.

Předávání klíče na forn-end je citlivá operace, proto se někdy vydává speciální TGT pro konkrétní front-end. Tj. použije samostatný `klíč-uživatel`.

18.10 Důvěra mezi říšemi

Kerberos umožňuje principálovi jedné říše se autentizovat vůči serveru v jiné říší pomocí vztahu důvěry mezi říšemi. Jelikož KDC může vydávat lístky pouze pro služby jeho říše, tak se použije trik – cizí říše se v naší říši registruje jako speciální principál. U tohoto principála je rovněž registrováno heslo, ze kterého se odvodí příslušný šifrovací klíč:

1. Uživatelův klient požádá KDC své říše (lokální KDC) o vydání lístku pro službu, která je registrována v cizí říši. Pokud si klient je vědom toho, tato služba je registrována v cizí říši, tak svoji žádost již označí příznakem NAME_CANONICALIZE.

2. Lokální KDC vytvoří lístek pro vydávání lístků, který odkazuje na KDC cizí říše. Jedná se o lístek relace pro cizí KDC. Pro šifrování klíče této relace využije heslo uvedené u speciálního principála cizí říše.
3. Klient postoupí tento nově vydaný lístek pro vydávání lístků KDC cizí říše.
4. KDC cizí říše vytvoří lístek relace pro požadovanou službu.
5. Klient uplatní lístek relace u požadované služby.
6. Volitelně může dojít k oboustranné autentizaci.

Registrace speciálního principála vytvoří vztah důvěry k cizí říši. Tento vztah je jednostranný, pokud má být oboustranný, pak obdobně musí cizí říše zaregistrovat speciálního principála naší říše.

Vztahy důvěry jsou tranzitivní, tj. Pokud říše A důvěřuje říši B a ta říši C, pak klient si může nechat vydat TGT v říši A, který odkazuje do říše B. Následně říše B mu vydá TGT, který odkazuje do říše C, jejíž KDC mu vydá lístek relace k požadované službě.

V případě, že je říší více, tak vytváření přímých vztahů důvěry mezi říšemi může vést k registraci velkého množství speciálních principálů, což může být obtížné administrovat. Kerberos v takovém případě doporučuje vytvořit stromovou strukturu vztahů důvěry mezi říšemi. Díky tranzitivitě vztahů důvěry můžeme mít pak vztah důvěry i s nejvzdálenější říší (např. skrze kořen stromu). Pokud bychom s nejvzdálenější říší potřebovali komunikovat často, pak nám nic nebrání vytvořit přímý vztah důvěry – tj. zkratku mezi větvemi stromu důvěry.

Problém je jak zjistit, jaké služby jsou dostupné v cizích říších. To není ale problém protokolu Kerberos. Microsoft to např. řeší komponentou Globální katalog, který mj. obsahuje SPN zajímavá pro všechny říše např. v interní síti firmy.

18.11 Časy

Maximální rozdíl v nastaveném času na klientovi, KDC a služby	5 minut
Maximální doba platnosti lístku	10 hodin
Maximální doba, po kterou je možné lístek obnovovat	7 dnů

I když 5 minut pro maximální rozdíl času uvedeného v žádosti a času KDC (resp. službu) se může zdát dlouhý, tak prakticky bez instalace služeb synchronizace času (např. NTP) je Kerberos jen těžko použitelný.

18.12 Struktura lístku

Datová struktura lístkuje následující:

Verze	5
Říše	Jméno říše *(Realm)*, KDC může vydávat lístky jen pro svou vlastní říši
Principál	Jméno služby *(Service Principal Name - SPN)*
Šifrováno	Příznaky (INITIAL, PRE-AUTHENT, …) – bitová mapa
	Klíč pro šifrování a dešifrování zpráv klienta a služby (Klíč-uživatel, Klíč-relace)
	Říše, ve které je uživatel registrován
	Jméno uživatele
	Seznam říší, které se podílely na autentizaci uživatele
	Čas, kdy se uživatel autentizoval
	Čas od kdy je lístek platný
	Čas do kdy je lístek platný
	Čas do kdy může být lístek obnovován
	Síťová adresa, ze které může být lístek použit (nemusí být uvedena, pak může být použit z libovolné adresy)
	Autorizační data: obsahují omezení použití lístku, ale Microsoft sem vkládá PAC - *Privilege Attribute Certificate*

Velice zajímavou položkou struktury lístku jsou příznaky lístku:

INITIAL	Lístek byl vydán na základě úvodní autentizace uživatele, tj. byl vydán AS – nikoliv TGT
PRE-AUTHENT	Tento příznak poskytuje informaci o úvodní autentizaci uživatele: úvodní autentizace obsahovala i před-autentizaci
HW-AUTHENT	Tento příznak poskytuje informaci o úvodní autentizaci uživatele: úvodní autentizace proběhla pomocí hardwarového zařízení.
INVALID	Lístek není platný. Používá se u listů vydaných s pozdějším datem platnosti (např. pro dávkové zpracování). Lístky vydané pozdějším datem musí být před použitím validovány KDC.
RENEWABLE	Lístek může být, před vypršením platnosti obnoven. Obnovený lístek obsahuje nový klíč relace. Lístky s tímto příznakem zpravidla též obsahují i čas, po kterém je již nelze lístek obnovit.
MAY-POSTDATE	Musí byt nastaveno v lístku pro vydávání lístků (TGT) v případě, že se požaduje vydání lístku s pozdějším datem platnosti.
POSTDATED	Lístek vydaný s pozdějším datem platnosti.
PROXIABLE	Lístek pro vydávání lístků (TGT) může být využit pro vydání lístku s příznakem PROXY.
PROXY	PROXY lístky se používají např., když klient chce vydat lístek, který následně předá front-end server, aby jeho jménem přistoupl na back-end server. Aby se ztížilo využívání kradených lístků, tak jsou lístky často vázány na síťovou adresu. PROXY lístek je tedy lístek relace, který bude požit z jiné síťové adresy, proto back-end může při použití PROXY lístků vyžadovat dodatečnou autentizaci fornt-end serveru.
FORWARDABLE	Lístek pro vydávání lístků (TGT) může být využit pro vydání lístku s příznakem FORWARDED. Jedná se obdobný mechanismus jako v případě lístku s příznakem PROXIABLE. Rozdíl je v tom, že klient pomocí lístku pro vydávání lístku (TGT) s příznakem FORWARDABLE předává svou identitu serveru, který si jeho jménem nechá vydat další lístek/lístky.
FORWARDED	FORWARDED lístek je možné využít obdobně jako PROXY lístek, tj. např. pro přístup webového serveru do databáze jménem klienta.
TRANSITED-POLICY-CHECKED	V případě, že lístek byl předáván mezi říšemi, pak KDC jednotlivých říší mohou mít pro toto předávání stanovena pravidla (politiky). Pokud byla tato pravidla při vydávání zkontrolována, pak je lístek označen příznakem TRANSITED-POLICY-CHECKED, pokud nikoliv, pak je označen příznakem DISABLE-TRANSITED-CHECK.
OK-AS-DELEGATE	Pokud klient deleguje pravomoc serveru jednat jeho jménem, tak může mýt pochybnost, jedná-li správně. Pomocí příznaku OK-AS-DELEGATE může KDC signalizovat klientovi, že server uvedený v lístku je politikou říše shledán vhodným pro delegaci.
NAME_CANONICALIZE	Jméno principála (serveru/služby), který vrací KDC ve své odpovědi (i v lístku) by

	se mělo shodovat se jménem principála, které bylo uvedeno v žádosti. Avšak zejména pokud je cílový server/služba v jiné říši, tak může KDC vrátit jiné jméno (alias) téhož principála. Příznak NAME_CANONICALIZE v žádosti vyjadřuje, že klient připouští vrácení i jiného jména principála než požadoval.

18.13 Vydání lístku

O vydání lístku se může žádat jak Autentizační server (AS) zprávou `KRB_AS_REQ`, tak i TGS zprávou `KRB_TGS_REQ`. AS pak odpoví buď zprávou `KRB_AS_REP` obsahující lístek nebo chybovou zprávou `KRB_ERROR`. `TGS` odpovídá zprávou `KRB_AS_REP`.

Žádost o vydání lístku (`KRB_AS_REQ`, `KRB_TGS_REQ`) má následující tvar:

Verze	5
Typ žádost	Žádost na AS (10), žádost na TGS (12)
Před-autentizace	Před-autentizační data
Tělo	Příznaky požadovaného lístku (bitová mapa)
	Jméno uživatele
	Jméno říše
	Jméno Služby *(Service Principal Name)*
	Čas od kdy má být lístek platný (volitelné)
	Čas do kdy má být lístek platný
	Čas do kdy má být lístek obnovitelný (volitelné)
	Nonce (náhodné číslo, které se kopíruje do odpovědi)
	Podporované šifrovací algoritmy (v preferovaném pořadí)
	Síťová adresa, ze které může být lístek akceptován (volitelné)
	Připojené lístky

Odpověď na žádost o vydání lístku (`KRB_AS_REP`, `KRB_TGS_REP`) má tvar:

Verze	5
Typ žádosti	Odpověď AS (11), odpověď TGS (13)
Před-autentizace	Před-autentizační data
Říše	
Jméno uživatele	
Lístek	
Šifrováno	Klíč
	Čas minulého požadavku klienta
	Nonce
	Čas, kdy vyprší platnost klientova hesla
	Příznaky (INITIAL, PRE-AUTHENT, …)
	Čas, kdy se uživatel autentizoval
	Čas od kdy je lístek platný
	Čas do kdy je lístek platný
	Čas do kdy může být lístek obnovován
	Síťová adresa, ze které může být lístek použit (nemusí být uvedena, pak může být použit z libovolné adresy)

18.14 Žádost o poskytnutí služby

V případě, že klient má již vystaven lístek pro poskytnutí služby, pak se může obrátit na příslušný server, aby mu služby poskytnul pomocí zprávy KRB_AP_REQ.

Žádost o poskytnutí služby obsahuje dvě zajímavé informace:

- Kontrolní součet z doprovázejících aplikačních dat, tj. z dat která klient zasílá na server.
- Dílčí tajný klíč. Tento klíč se může pak použít např. pro zabezpečení následné komunikace.

Verze	5
Typ žádosti	14
Volby	Např. je požadována oboustranná autentizace, tj. je požadována odpověď na žádost o poskytnutí služby
Lístek	
Jméno uživatele	
Šifrováno	Verze = 5
	Jméno říše
	Jméno klienta
	Kontrolní součet z doprovázejících aplikačních dat (volitelné)
	Čas
	Dílčí tajný klíč (volitelné)
	Pořadové číslo (volitelné)
	Autorizační data (volitelné)

Odpověď na žádost o poskytnutí služby je volitelná, používá se v případě oboustranné autentizace. Odpověď KRB_AP_REP má následující strukturu:

Verze	5
Typ žádosti	15
Šifro-váno	Čas
	Dílčí tajný klíč (volitelné)
	Pořadové číslo (volitelné)

18.15 Kerberos ve Windows

Stanice při začlenění do domény Windows (=říše) si zaregistrují dlouhodobý klíč (heslo), takže z hlediska Kerbera je pak přihlášení uživatele k účtu na stanici ve své podstatě autentizací k službě. Obdobně probíhá i přihlášení k doménovému účtu.

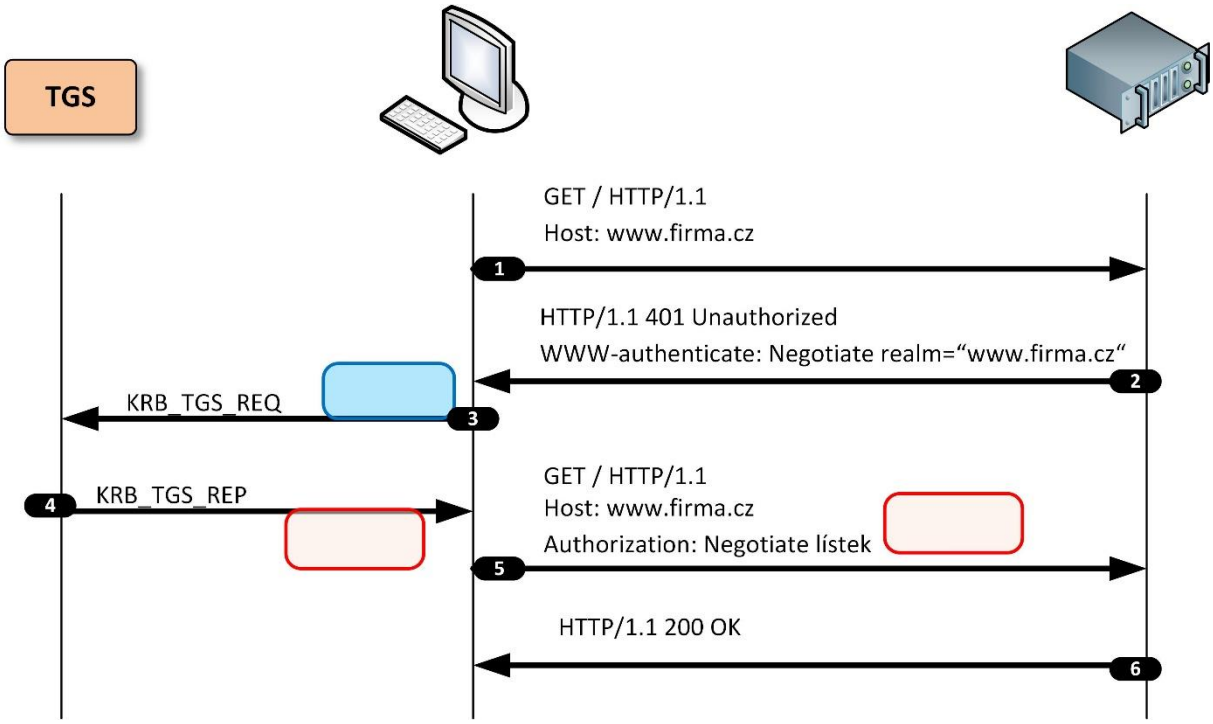
Kerberos je zásadně jen autentizační protokol. Microsoft jej ale rozšířil o autorizaci. Konkrétně tak, že do TGT lístků vydávaných Autentizačním serverem, do položky Autorizační data vkládá strukturu PAC *(Privilege Attribute Certificate)*. Tato struktura obsahuje data, která budou základem bezpečnostních kontextů všech procesů spuštěných takto autentizovaným uživatelem. PAC může být též součástí lístku relace, čímž se předají autorizační informace službě. PAC mj. obsahuje: SID, členství uživatele ve skupinách a další bezpečnostní atributy uživatele. Struktura PAC je šifrována a zabezpečena kontrolním součtem, tak jak bylo o šifrování a kontrolním součtu výše zmíněno.

Na stanici s operačním systémem Windows, která je v doméně Active Directory s podporou Kerbera si může uživatel, v příkazovém řádku, vypsat své lístky příkazem klist (doporučuji vyzkoušet na vaší stanici Windows – zjistíte např., že služba krbtgt reprezentuje TGS).

18.16 UNIX a Windows

UNIXový svět používá protokol Kerberos ve své čisté podobě. Takže díky rozšíření Kerbera realizovaných Microsoftem je snadnější začlenit server/službu z ne-Windowsového prostředí do Windowsového prostředí než naopak.

Začleňování UNIXových serverů do prostředí Windows je dnes běžné. Je ale vždy si třeba uvědomit, že heslo serveru musí být uloženo jak v ActiveDirectory, tak musí být dostupné i UNIXovému serveru, který nemá přístup do ActiveDirectory. Tj. musí se konfigurovat dvakrát. Microsoft používá pro tyto účely soubor Keytab a příkaz Ktpass.



Obrázek 18.10 Autentizace na web protokolem SPNEGO

18.17 Autentizace na Web

Pro autentizaci na web protokolem Kerberos se nejčastěji používá protokol SPNEGO (RFC-4559). Tato autentizace se používá, jak pro autentizaci uživatelů jménem a heslem (metoda Basic), tak i pro autentizaci protokolem Kerberos nebo NTLM (metoda Negotiate), která je znázorněna na následujícím obrázku. (Protokol SPNEGO podporuje i další metody, jako např. Digest, která se používá např. také pro autentizaci heslem, ale asi známější je v prostředí VoLTE, kde se používá autentizace algoritmem AKA.)

Starší verze protokolu TLS podporovaly i protokolové sady s autentizací protokolem Kerberos. V novějších verzích se tyto sady již nevyskytují. Zajímavé je, že protokolové sady s protokolem Kerberos zůstaly v protokolu DTLS (datagramové TLS).

18.18 Co dále?

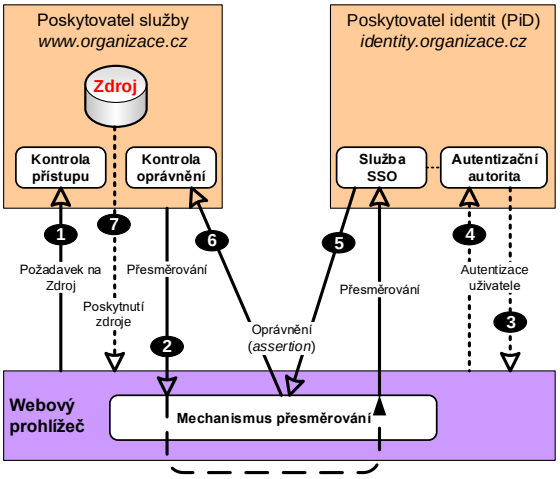
Nejslabším článkem protokolu Kerberos jsou slabá hesla. Používá se dokonce rčení: Kerberos je tak bezpečný, jak silná jsou uživatelská hesla (jakou mají entropii).

Protokol Kerberos je koncipován jako autentizační protokol určený pro intranety. Rozšíření Microsoft to ještě zvýraznily, když do lístku doplnily bezpečnostní atributy.

Zejména pro využití v internetu se problematika zobecňuje a hovoří se o tzv. federaci identit, která se na internetu dnes realizuje např. pomocí protokolů SAML, JWT, Oauth2 a OpenID Connect. Tyto protokoly neposkytují takový komfort, jako Kerberos. Jsou ale dnes považovány za bezpečnější. Na druhou stranu protokol Kerberos se používá už 30 let. Uslyšíme za 30 let o protokolech SAML, JWT, Oauth2 a OpenID Connect?

19 Federace identit

Protokol Kerberos řeší i problém důvěry mezi říšemi, tj. řeší i problém jak se lístkem vydaným v jedné říši prokázat v jiné říši.



Obr. 19.1 SAML je postaven na přesměrování

V současné době se, pro federaci identit (kromě protokolu Kerberos) používají zejména dva standardy:

- *Security Assertion Markup Language* (SAML) – nyní ve verzi 2.
- *Open Authentication* (OAuth) – nyní ve verzi 2.

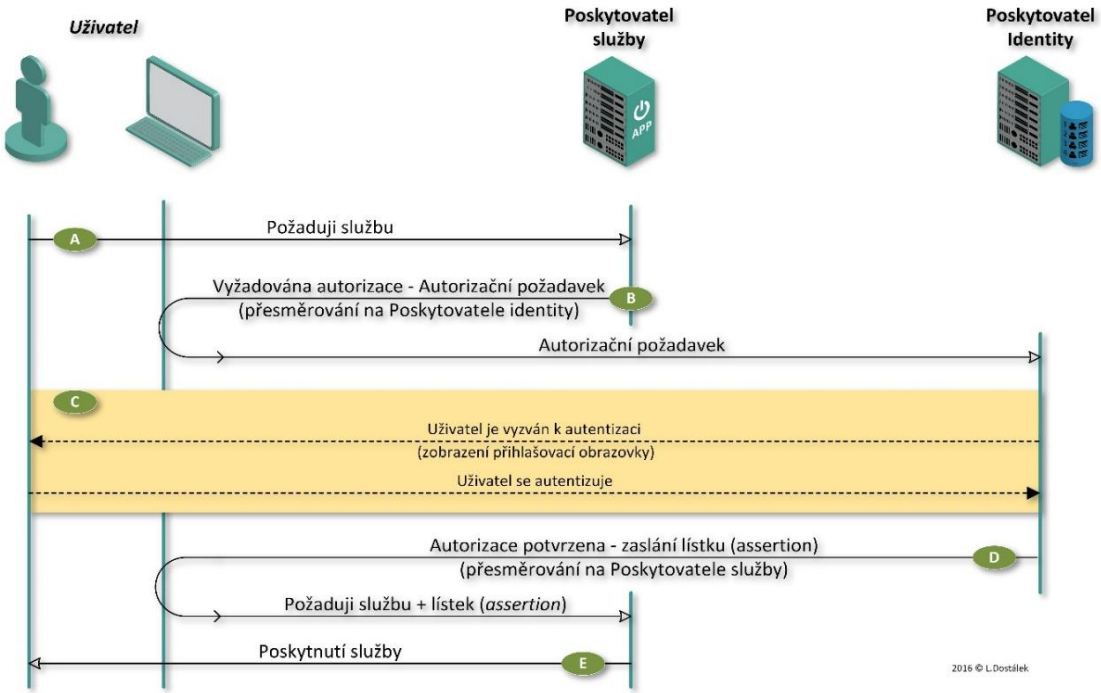
19.1.1 SAML

SAML (*Security Assertion Markup Language*) řeší problém federace tak, že odděluje poskytovatele služby (tj. poskytovatele zdroje informace o kterou má subjekt zájem) a poskytovatele identity

Uživatel z webového prohlížeče žádá poskytovatele služby o poskytnutí nějaké služby. Poskytovatel služby vyžaduje autentizaci, tak pomoci přesměrování přesměruje požadavek na poskytovatele identit (Obr. 19.1). Poskytovatel identit autentizuje uživatele (není součástí protokolu) a v případě kladného výsledku vydá o proběhlé autentizaci potvrzení („lístek“). Na závěr ještě uživatele vybaveného potvrzením o autentizaci přesměruje na poskytovatele služby. Součástí potvrzení mohou být i autorizační data.

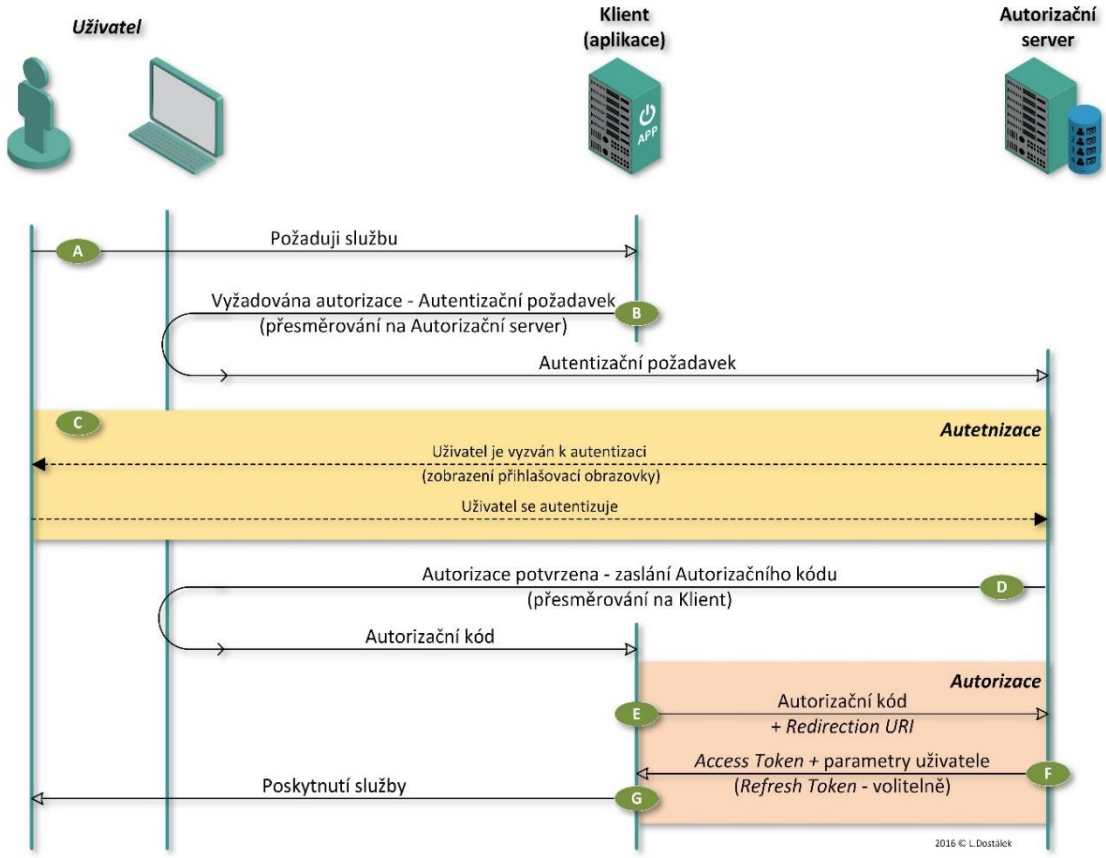
Poskytovatel identity tedy provádí autentizaci subjektu a případně ji doplní i autorizací. Výsledkem autentizace je vydání lístku[1] (*Assertion*), na základě kterého poskytovatel služby poskytne/neposkytne příslušný zdroj. Federace spočívá v tom, že Poskytovatel identit poskytuje lístky různým poskytovatelům služeb (obr. 19.2).

SAML sám je jen manipulační jazyk, který popisuje lístek (*Assertion*). Tj. jak samotná autentizace, tak



obr. 19.2 SAML: Poskytovatel identity stvrzuje identitu uživatele poskytovateli služby

[1] Někdy se též používají termíny oprávnění, tvrzení, token atp.



obr. 19.3 Příklad dialogu protokolu OAuth 2.0

mechanismus přesměrování[2] znázorněný na obr. 19.2 jsou mimo specifikaci tohoto standardu. Závisí na konkrétní implementaci.

SAML *Assertion* je XML struktura, která zpravidla obsahuje i elektronický podpis (XML signature), kterým je stvrzena pravost. Navíc může být tato struktura i šifrována.

19.1.2 JWT

Manipulační jazyk SAML je velice obecný, ale díky své obecnosti je jednak složitý a jednak výsledný lístek příliš komplikovaný, což někdy vyvolávalo technické obtíže. Autentizační informace se proto dnes častěji nepopisují ve tvaru SAML, ale pomocí *JavaScript Object Notation* (JSON). Vznikl tak standard JSON Web Token (JWT).

Na rozdíl od SAML struktura JWT zpravidla není elektronicky podepsána ani šifrována.

19.1.3 OAuth 2.0

Jak SAML, tak i JWT popisují jen lístek, který vydává poskytovatel identity subjektu, aby se jím prokázal poskytovateli služby. Protokol, kterým dojde k této komunikaci, je mimo tyto standardy (tj. není součástí těchto standardů).

OAuth 2.0 je protokol, který tento problém řeší, tj. popisuje tuto komunikaci. OAuth 2.0 umožňuje obdobnou autentizaci jako na obr. 19.2. Umožňuje i jiné dialogy - na obr. 19.4 je příklad dialogu protokolu OAuth 2.0, kdy je rozdělen dialog do dvou fází:

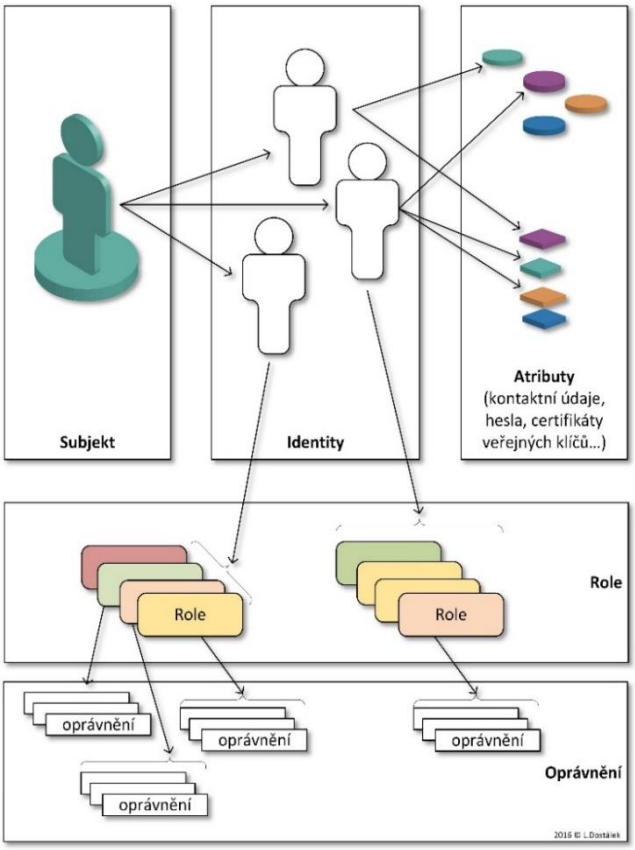
1. Autentizace, jejíž výsledkem je získání Autorizačního kódu, který může být náhodný, a tak nezadat šanci útočníkovi útočícímu na uživatelův počítač zneužít lístek zaslaný Autorizačním serverem.
2. Autorizace, kdy Klient (aplikace) získá přístupový lístek s oprávněními poskytnout

[2] Zpravidla se využívá mechanismus přesměrování protokolu HTTP

uživateli příslušnou službu. Lístek se zde nazývá *Access Token*. Klient může získat i tzv. *Refresh Token*, sloužící k obnovení lístku.

19.2 RBAC model

Role-Based Access Control (RBAC) model předpokládá, že subjekt má v rámci nějaké oblasti/domény/říše (např. organizace) jednu nebo více identit (Obr. 19.5). Každá jeho identita má konkrétní atributy (kontaktní údaje, hesla, certifikáty veřejných klíčů atp.). Důležité ale je, že konkrétní oprávnění pro přístup a práci s aktivy nejsou přímo vázaná na identitu, ale na role. Tj. identitám jsou přiřazeny role a teprve na role jsou navázána oprávnění. Při změně role, tak automaticky dojde ke změně oprávnění. Roli si můžeme představit např. jako pozici v organizaci (většinou v praxi jedné pozicí odpovídá více rolí). Role může být ale třeba občan při styku občana se státní mocí.

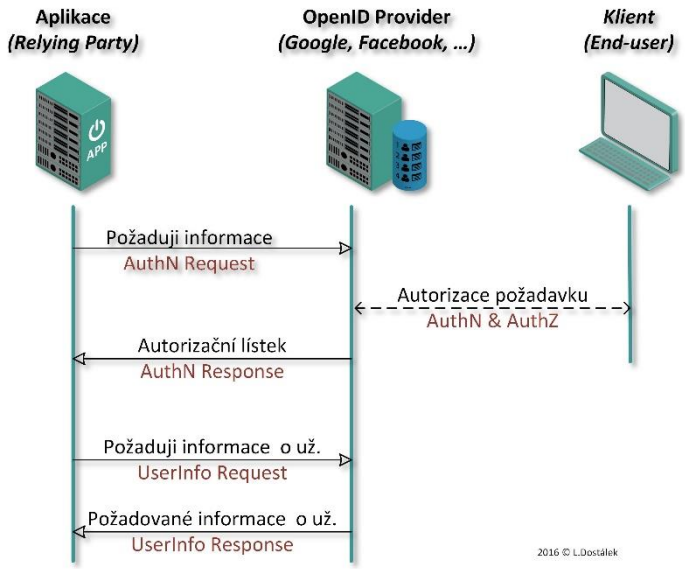


Obr. 19.5 RBAC model

19.3 OpenID Connect

Informace o uživateli udržuje zpravidla ověřovatel. V případě, že využíváme federaci identit, pak je užitečné získat atributy ověřené identity od prvotního ověřovatele. OpenID Connect (obr. 19.6) je protokol, který umožňuje získání atributů identity od původního ověřovatele.

Přiklad: Pro ověření do aplikace budeme využívat přihlášení do systému Facebook (v systému Facebook máme odkaz do uvedené aplikace). V případě, že uživatel přejde na tento odkaz, díky federaci identit se akceptuje autentizace ze systému Facebook do uvedené aplikace. Pro založení uživatele v aplikaci potřebujeme jeho atributy. Ty získáme přes protokol OpenID Connect.



obr. 19.4 OpenID Connect