MODERNÍ KRYPTOGRAFICKÉ METODY

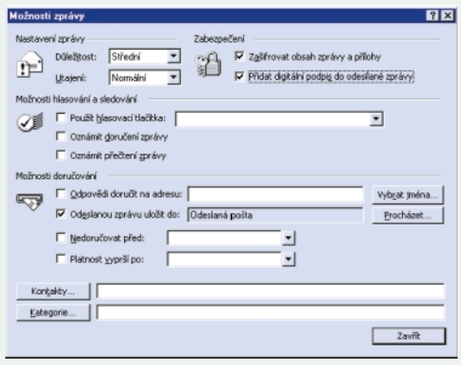
Bude nás podepisovat RSA?

Na našem trhu se brzo objeví zahraniční i domácí prostředky a služby pro realizaci elektronického podpisu v praxi. Lze očekávat, že řada z nich bude založena na algoritmu RSA, a proto se nyní věnujeme popisu standardu PKCS#1 pro jeho použití. Ukážeme si konkrétní realizaci RSA na elektronickém podpisu a na šifrování klíčů a seznámíme se také s některými triky a pojmy, se kterými se budeme u RSA setkávat v čipových kartách nebo jiných prostředcích pro elektronický podpis.

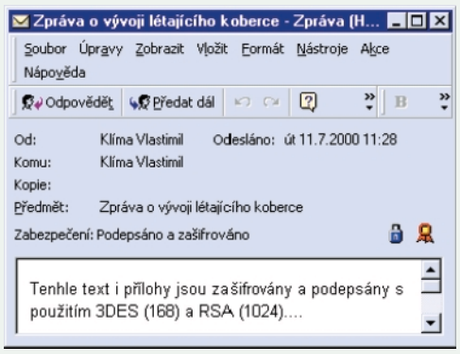
VÝMĚNA KLÍČŮ A PODPIS DAT

Pokud si zvolíte algoritmus RSA pro elektronický podpis (dále jen podpis) podle přijatého zákona o elektronickém podpisu, budete se nutně muset seznámit se ***standardem PKCS#1*** – ten totiž definuje operaci zašifrování a odšifrování bloku dat algoritmem RSA. Proto je PKCS#1 základním kamenem ostatních norem PKCS také z bezpečnostního hlediska. Jak jsme uvedli minule, asymetrické šifry se využívají v zásadě ke dvěma účelům, a to k výměně symetrických šifrovacích klíčů (šifrování) a k podepisování dat (podpis). Dále se podíváme, jak se tyto činnosti dělají pomocí algoritmu RSA.

Začneme příkladem. Dejme tomu, že už máme k dispozici svůj podpisový klíč i certifikát a v našem programu pro práci s elektronickou poštou (poštovní klient) chceme podepsat nebo zašifrovat odesílaný e-mail (nebo obojí současně). Ve většině případů jen zaškrtneme políčko označené zpravidla „Zašifrovat“ nebo „Podepsat“ (viz obr. 1) nebo klepneme na nějakou ikonu. Poštovní klient pak naše přání splní, k čemuž volá různé pomocné funkce, včetně kryptografických. Poštovní klienty Microsoftu a Netscape předloží nou zprávu zpracují pomocí formátu S/MIME *(Secure Multipart Internet Mail Extensions)*. S/MIME použije k podpisu i šifrování zprávy formát standardu PKCS#7 a ten se řídí standardem PKCS#1. PKCS#1 obstará přípravu a formát vstupních i výstupních dat pro algoritmus RSA.



Obr. 1. Volba možnosti zašifrovat a podepsat e-mail

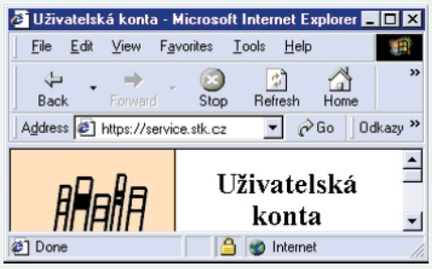


Obr. 2. Příjem zašifrovaného a podepsaného e-mailu indikují ikonky zámečku a pečetě.

ŠIFROVÁNÍ S SSL

Budeme-li mít své bankovní konto přístupné prostřednictvím internetu, budeme asi chtít, abychom s ním mohli manipulovat jen my. V tomto případě uvítáme spojení zabezpečené prostřednictvím protokolu SSL *(Secure Sockets Layer)*, který je nejpoužívanějším aplikačním protokolem pro šifrování dat na internetu. Že se jedná právě o toto spojení, poznáme z adresy příslušného serveru – začíná nikoli *http://...*, ale ***https://...*** , viz obr. 3. Písmeno „s“ na konci znamená, že mezi vrstvu TCP/IP a aplikační protokol HTTP je vložen právě bezpečnostní protokol SSL, který umí (pokud je správně nakonfigurován) zajistit:

- **vzájemnou autentizaci** obou komunikujících stran, tj. nás (jako klienta) a serveru: server ví, že se na něj dobýváme právě my, a my víme, že je to server právě naší banky;



Obr. 3. Adresa zabezpečeného serveru (https://) a ikona spojení zabezpečeného protokolem SSL

- **integritu dat** (to, co vidíme v prohlížeči, je skutečně stav našeho konta, a nikdo tuto informaci nemohl změnit při jejím putování internetem);
- **šifrování dat** zajišťující soukromí, takže uskutečněnou komunikaci nelze na internetu „odposlouchávat" v otevřené podobě (neuspěje ani poskytovatel připojení, ani útočník).

Je-li spojení zabezpečené protokolem SSL úspěšně navázáno, poznáme podle ikonky zámečku, která se objeví v liště programů (obr. 3). Pokud se během úvodní fáze protokolu SSL obě komunikující strany dohodnou na použití algoritmu RSA pro výměnu klíčů, pak k výměně šifrovacího klíče pro šifrování další komunikace je použit právě standard PKCS#1.

Vidíme tedy, že jak při **podpisu**, tak i při **šifrování klíčů** se v obou případech (e-mail, SSL) nakonec použije RSA podle PKCS#1. Nyní se tedy této operaci věnujeme podrobněji (a zatím ponecháme stranou další bezpečnostní a aplikační aspekty, jako například kde je uložen a jak je chráněn privátní klíč, jak je zajištěna infrastruktura veřejných klíčů apod.).

RSA PRAKTICKY

Popis RSA i s příklady jsme v Chipu už vysvětlili (příslušný článek z Chipu 4/95 je k dispozici také na internetu, viz infotipy); zopakujme jen, že základní operace RSA je $\mathbf{c = m^e \bmod n}$ pro šifrování a $\mathbf{m = c^d \bmod n}$ pro odšifrování. Nyní se soustředíme na některé pojmy, s kterými se můžete u RSA setkat.

V první řadě je to délka modulu. **Nejpoužívanější délka modulu RSA je a bude 1024 bitů**. V Česku je to sice zatím 512 bitů, protože většina uživatelů ještě nepoužívá software se silnou kryptografií (od uvolnění vývozu uplynula příliš krátká doba), od kratších modulů (512 a 768) se ale ustupuje z bezpečnostních příčin. Modul 512 bitů byl už faktorizován a 768 bitů je „na dostřel". Naproti tomu moduly delší (2048, 4096), poskytující nadstandardní bezpečnost, se zase nerozšířily, protože výpočty s nimi jsou v současné době ještě stále pomalé.

Dalším pojmem, na který můžeme narazit zejména u čipových karet, je zkratka **CRT** *(Chinese Remainder Theorem)*. Je to matematická věta (tzv. čínská věta o zbytku), pomocí níž se (v čipových kartách i v softwaru) dosahuje kvalitativně lepších časů na provedení operace RSA s tajným klíčem, což je právě případ, kdy něco elektronicky podepisujeme. Podobně tzv. *Montgomeryho metoda* (nebo redukce) je postup urychlující základní operace modulárního násobení, které RSA používá mnohokrát za sebou.

OBSAH STANDARDU PKCS # 1

U algoritmu RSA nejde jen o funkci modulárního mocnění, ale pro praktické využití se musí definovat ještě formát dat a jejich doplňování a dodržet určitá pravidla pro generování klíčů. PKCS#1 z velké části hovoří vlastně o tom, jak se zpracovávaná data doplní do plného bloku RSA. Klíče, které se šifrují, i haše, které figurují u elektronického podpisu, v praxi totiž vyplňují jen malou část bloku RSA. Je také potřeba bitové řetězce převést na čísla, aby se s nimi mohla provést operace $\mathbf{x^y \bmod z}$, a po jejím provedení zase výsledné číslo převést zpět na bitový řetězec (je to typ BITSTRING podle normy ASN.1 – k ní se ještě vrátíme v některém dalším dílu).

PKCS#1 proto definuje tyto datové konverze, dále uvádí formáty pro ukládání veřejných a tajných klíčů a ještě zavádí tzv. objektové identifikátory podle normy ASN.1 apod. Verze 1.5 standardu PKCS#1 byla první použitelnou verzí a byla publikována 1. 11. 1993. Přestože ji od 1. 10. 1998 nahradila verze 2.0, je umožněna zpětná kompatibilita. **Verze 1.5 je proto stále naprosto převládající v poštovních klientech i v internetových prohlížečích.** Později uvidíme, že pro podpis dat je tento standard z bezpečnostního hlediska zatím v pořádku, ale pro šifrování už ne. Byla totiž nalezena skulina, jak formát dat pro šifrování klíčů využít k úspěšné kryptoanalýze.

OZNAČENÍ, SYMBOLY A KONVERZE

V připojené tabulce uvádíme základní označení, která dále používáme. Připomeňme, že *oktety* jsou osmice bitů, tedy vlastně bajty, ale protože toto označení se používá i v souvisejících normách (ASN.1), budeme se ho držet. Pro označování řetězců bitů nebo řetězců oktetů budeme používat velká písmena, pro čísla písmena malá.

Konverzi mezi čísly a oktety musíme nadefinovat z příčin, které jsme uvedli výše, ale je to jednoduché. Číslo se při konverzi na řetězec oktetů jen eventuálně zleva doplní nulovými bity tak, aby mělo binární vyjádření zarovnané na osmice bitů (oktety), a naopak řetězec oktetů se obvyklým způsobem převede na číslo tím, že jeho oktety nejvíce vlevo se budou chápat jako bajty s nejvyšší vahou. Formálně se tyto procedury nazývají I2OSP *(Integer-to-Octet-String Primitive)* a OS2IP *(Octet-String-to-Integer Primitive)*.

SE ZKRATKOU CRT SE **SETKÁTE U ČIPOVÝCH KARET** – ZNAMENÁ „ČÍNSKOU VĚTU O ZBYTKU", KTERÁ URYCHLUJE **ELEKTRONICKÝ PODPIS**.

Symboly a označení

Symbol	Poznámka, význam
ab	konkrétní oktet, osmibitový řetězec (hexadecimálně); není kurzivou
BT	tzv. typ bloku *(Block Type)*, nabývá hodnoty 01 nebo 02
D	bitový řetězec *(Data)*
EB	řetězec oktetů *(Encryption Block)*, připravený bezprostředně ke konverzi na číslo a poté k operaci zašifrování RSA
ED	výsledek operace RSA po konverzi na řetězec oktetů *(Encrypted Data)*
M	původní zpráva k podpisu *(Message)*
MD	hašovací kód M *(Message Digest)*
PS	doplňující řetězec *(Padding String)*
X || Y	zřetězení X a Y

Práce s daty podle PKCS#1, ver. 1.5

Data *D*, která vstupují do algoritmu RSA, mají obvykle délku do 40 oktetů (320 bitů, jsou to klíče nebo haše), takže je nutné je doplnit do zvolené délky bloku (modulu) RSA; tuto délku označme *k* (oktetů). Doplněný blok označíme *EB*; v PKCS#1, ver. 1.5, je definován jako řetězec *k* oktetů podle vztahu $EB = 00 \,||\, BT \,||\, PS \,||\, 00 \,||\, D$.

Typy bloků 01 a 02

Kromě vlastních dat *D* vystupuje v zápisu pro *EB* ještě **separátor** (oktet 00), **doplňující řetězec** několika oktetů *PS (padding string)*, dále jeden oktet *BT* a vedoucí oktet 00. Počet oktetů doplňujícího řetězce *PS* se volí tak, aby celková délka *EB* byla požadovaných *k* oktetů; z bezpečnostních příčin se požaduje délka *PS* alespoň 8 oktetů. Vedoucí oktet 00 (v *EB* nejvíce vlevo) je povinně zaveden proto, aby při převodu *EB* na číslo (OS2IP) byl nejvýznamnější bajt tohoto čísla vždy nulový. Zpracovávané číslo je tak vždy menší než modul RSA, což je nutné pro správnost odšifrování.

Důležitou roli hraje v zápisu pro *EB* oktet *BT (block type)*, který určuje **typ** příslušného bloku. Může nabývat hodnot 00 (nepoužívá se) nebo 01 a 02 (kompatibilita s formátem PEM podle RFC 1423, důležité dříve). Typ bloku 01 je určen pro **podpis dat** a typ 02 pro **šifrování klíčů**.

Doplňování bloku typu 01

V bloku typu 01 (podpis) je *PS* tvořen pouze **stejnými oktety s hodnotou FF**. Vlastní data *D*, což je zde hašovací kód *MD* podepisované zprávy *M*, se zde ale navíc doplňují ještě konstantním identifikačním řetězcem. V notaci ASN.1 je to tzv. *DigestAlgorithmIdentifier* a jeho hodnota je odvozena od toho, jaká hašovací funkce se použije k hašování zprávy *M*. Jsou definovány identifikátory pro MD2, MD5 a SHA-1. Více o uvedených hašovacích funkcích viz infotipy.

Doplňování bloku typu 02

V tomto případě je *PS* tvořen jakýmikoliv **náhodnými nenulovými oktety**. Nenulovost má samozřejmě umožnit jejich jednoznačné odlišení od významových dat pomocí nulového separátoru. Jen pro zajímavost si všimněme, že kvůli náhodnosti doplňovaných bajtů budou v bloku typu 02 (šifrovací klíče) stejná data *D* mít pokaždé jiný šifrový obraz. Přestože se to zdá jako velmi silné opatření, právě v realizaci myšlenky doplnění zcela náhodnými daty je skryta možnost luštění. Je to o to horší, že se jedná o šifrovací klíče, kterými se šifrují data přenášená v kanálu.

Šifrujeme a podepisujeme...

Jakmile je připraven plný blok *EB*, pomocí procedury OS2IP ho převedeme na číslo, aplikujeme na něj algoritmus RSA buď s veřejným, nebo tajným exponentem a obdržené číslo převedeme pomocí procedury I2OSP zpět na oktetový řetězec *(ED)*. Ten pak tvoří výsledek celé operace. Na straně příjemce se na obdržený oktetový řetězec zavolá algoritmus RSA s odpovídajícím párovým klíčem a u výsledku se zjistí, zda obdržený formát dat odpovídá formátu daného typu bloku. Kontroluje se nulový vedoucí oktet, oktet *BT*, vlastnosti doplňku *PS*, přítomnost separátoru a délka vlastních dat. Při verifikaci podpisu se navíc kontroluje identifikátor hašovacího algoritmu a obdržená *MD* se také porovná s hašovací hodnotou vypočtenou z přijatých podepsaných dat *MD'*.

Jak je to s bezpečností

V současné době je v neamerických verzích řady programů stále ještě používán modul RSA 512 bitů. Vývoz silné kryptografie z USA, umožňující používat modul 1024 bitů, byl částečně uvolněn letos v lednu a pro ČR zcela v červenci, ale do praxe se tato změna dosud příliš nepromítla. Až si nainstalujeme příslušné programy nebo „service packy“, můžeme s komerčními prohlížeči a poštovními klienty díky tomuto uvolnění už dnes dosáhnout velmi slušné „lidové“ bezpečnosti.

Abychom mohli komunikovat na vyšší bezpečnostní úrovni, musí k tomu ovšem navíc existovat bezpečná infrastruktura veřejných klíčů. Velmi důležitá je také konfigurace příslušných programů, důvěryhodnost a vlastnosti certifikátů a nakonec ochrana tajných klíčů uživatele v systému. Řady těchto aspektů se týká nedávno přijatý zákon o elektronickém podpisu, a v Chipu se proto brzy chceme věnovat také hlubšímu pohledu na jeho příslušná ustanovení.

Závěr

Minule jsme se seznámili s řadou standardů PKCS, která obsahuje nejpoužívanější normy v oblasti asymetrických systémů. V tomto dílu jsme se zabývali základem této řady, PKCS#1, a ukázali jsme, jak se podle verze 1.5 této normy vytváří podpis nebo šifrují klíče algoritmem RSA. (Verzí 1.5 jsme se zabývali proto, že je stále dominantní a na novější 2.0 se ještě všeobecně nepřešlo.) V příštím dílu si ještě povšimneme jedné její slabiny a ukážeme, jak se jí bránit.

Vlastimil Klíma (v.klima@decros.cz)

POKUD SI ZVOLÍTE RSA, **BUDETE SE MUSET SEZNÁMIT** SE STANDARDEM PKCS#1.

VERZE 1.5 PKCS#1 JE **DOMINANTNÍ V POŠTOVNÍCH KLIENTECH** I INTERNETOVÝCH PROHLÍŽEČÍCH.

infotipy

Všechny použité pojmy jsou podrobně vysvětleny v následujících článcích.
Naleznete je také na adrese:
- *www.decros.cz/Security_Division/Crypto_Research/archiv.htm*

nebo
- *ftp://ftp.decros.cz/pub/Archiv/Publications/*

Jsou zde uvedeny pod mnemotechnickým označením
časopis-rok-měsíc-strana(od)-strana(do).ext.
O principech asymetrických šifer:
V. Klíma: Revoluce v šifrování!, Chip 2/95, str. 126 – 128.
RSA - podstata, bezpečnost, luštitelnost:
V. Klíma: Šifrový šampión, Chip 4/95, str. 136 – 138.
Kreknutí RSA-129:
V. Klíma: Internet a RSA, Chip 6/95, str. 174 – 175.
Kreknutí RSA-155:
T. Rosa: Jde to i bez TWINKLU, Chip 10/99, str. 30 a 34.
O lušticím zařízení TWINKLE:
T. Rosa: Na to vezmi LED!, Chip 8/99, str. 40 – 43 a 9/99, str. 34 – 37.
O slabosti exponentu e=3 i jiných útocích:
T. Rosa: Když se tesař utne, Chip 6/97, str. 160 – 163.
O hašovacích funkcích MD2, MD4, MD5 a SHA-1:
V. Klíma: Výživná haše, Chip 3/99, str. 40 – 43;
V. Klíma: Jak se melou data, Chip 4/99, str. 44 – 46.
Čínská věta o zbytku pro RSA:
[QC82] J. – J. Quisquater, C. Couvreur: Fast decipherment algorithm for RSA public-key cryptosystem, Electronics Letters, 18(21):905–907, October 1982.
Standardy PKCS:
- *http://www.rsasecurity.com/rsalabs/pkcs/*