PKCS#7

I šifra musí mít formát

Pokud máme k dispozici algoritmy pro šifrování dat a zaručený elektronický podpis, nic nám už nebrání podepisovat a šifrovat e-maily, soubory atd. Až na jednu drobnost: chceme-li být kompatibilní s ostatním světem, musíme dodržet dohodnutý formát. V tom nám pomůže standard PKCS#7.

Srodinou standardů PKCS (Public Key Cryptography Standards) jsme vás poprvé seznámili v Chipu 8/00 (viz infotipy). Tyto standardy se týkají zejména kryptografie s veřejným klíčem, ale také symetrických šifer, hašovacích funkcí, zpracování passwordů apod. Standard PKCS#7, o němž bude řeč nyní, definuje potřebné formáty zašifrovaných i podepsaných dat a tzv. digitální obálku. Je také nejrozšířenějším formátem pro zabezpečení elektronické pošty na internetu, neboť je základním stavebním kamenem formátu S/MIME, který využívají oba nejdůležitější výrobci – Microsoft i Netscape.

Verze a datové typy PKCS#7

PKCS#7 hraje v rodině těchto standardů důležitou roli a je těsně propojen s ostatními členy PKCS, neboť definuje syntaxi (formát) zpráv, na něž jsou aplikovány důležité kryptografické funkce. V současné době je platná verze 1.5, které se zde budeme věnovat – ale ihned poznamenejme, že už byl vydán neoficiální seznam změn, s nimiž se počítá pro verze 1.6 a 2.0; ještě se k nim vrátíme v závěru článku.

PKCS#7 definuje šest datových typů (struktur, formátů dat) popsaných pomocí jazyka ASN.1 (psali jsme o něm v Chipu 12/00, viz infotipy) – společně s jejich objektovými identifikátory je můžete vidět v rámečku 1. Jakýmsi zobecněním všech šesti typů dat je struktura *ContentInfo*, která vytváří jednotný rámec pro všech šest typů. Ty se tedy navenek tváří jako *ContentInfo* a teprve uvnitř této struktury je objektovým identifikátorem specifikován a uložen každý ze šesti základních typů (viz rámeček 1).

‹1› Základní datové typy PKCS#7 a jejich objektové identifikátory

data	OBJECT IDENTIFIER ::= { pkcs-7 1 }
signedData	OBJECT IDENTIFIER ::= { pkcs-7 2 }
envelopedData	OBJECT IDENTIFIER ::= { pkcs-7 3 }
signedAndEnvelopedData	OBJECT IDENTIFIER ::= { pkcs-7 4 }
digestedData	OBJECT IDENTIFIER ::= { pkcs-7 5 }
encryptedData	OBJECT IDENTIFIER ::= { pkcs-7 6 }

přičemž pkcs-7 je také objektový identifikátor, rovný hodnotě
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 7 }

Zastřešující typ nad všemi šesti uvedenými datovými typy je

```
ContentInfo ::= SEQUENCE {
        contentType   ContentType,
        content       [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL }
```

kde
ContentType ::= OBJECT IDENTIFIER (viz výše)

Univerzalita

Z mnemotechnických názvů datových typů je vidět, že kromě základního typu *Data,* který obsahuje nijak neupravovaná data (je definován jako Data::= OCTET STRING), je možné definovat data šifrovaná *(encrypted)*, podepsaná *(signed)*, data digitálně obalená *(enveloped data)* atd. Podstatné je, že tyto typy (digitální obálky) lze libovolně vnořovat do sebe, což umožňuje vytvářet takové struktury, které vyhoví mnoha kryptografickým požadavkům. Lze ale využít i nejjednodušší formu datového typu. Například k rozesílání seznamů certifikátů *(Certificate List, CL)* a zneplatněných certifikátů *(Certificate Revocation List, CRL)* se využívá formát „podepsaná data“, kde množina podepisujících je prázdná. Syntaxe také umožňuje k datům připojovat různé atributy, jako například čas podpisu, umožnit více podepisujících apod.

> PKCS#7 je perspektivní **formát** pro zpracování a přenos kryptograficky upravených dat.

Často využívané typy

Nejprve si připomeneme datové typy, se kterými jsme se už seznámili v předchozích dílech, ale nyní je budeme potřebovat.

AlgorithmIdentifier – identifikátor algoritmu definovaný přímo v X.509, jehož pomocí se definuje kryptografický algoritmus (symetrický, asymetrický, podpisové schéma apod.) a jeho parametry. *Attribute* – datový typ definovaný v X.501, který je určen typem atributu *(attribute type)* a jednou nebo více hodnotami atributu *(attribute values)*. Typ atributu je specifikován objektovým identifikátorem (o nich jsme psali v čísle 12/00, viz infotipy).

Certificate – datový typ definovaný v X.509 (blíže viz Chip 12/00), který svazuje jméno subjektu *(distinguished name)* s jeho veřejným klíčem. Obsahuje také jednoznačné jméno vydavatele certifikátu (certifikační autorita, CA), sériové číslo *(CertificateSerialNumber)*, identifikátor algoritmu a dobu platnosti.

CertificateRevocationList – seznam certifikátů zneplatněných před přirozeným vypršením doby jejich platnosti (CRL). Obsahuje jméno vydavatele *(CA)*, čas vydání CRL, čas vydání následujícího CRL a seznam sériových čísel a zneplatněných certifikátů (včetně času zneplatnění), to vše pak podepsáno CA.

Name – často používaný typ definovaný v X.501; představuje jednoznačné jméno v adresáři X.500. *IssuerAndSerialNumber* – definován jako SEQUENCE {issuer Name, serialNumber CertificateSerialNumber} čili jako dvojice „jméno CA a sériové číslo certifikátu“; jednoznačně tak ukazuje na certifikát a na údaje v něm uvedené, tj. na jméno a klíč subjektu.

‹2› Definice datového typu

```
SignedData ::= SEQUENCE {
  version           Version,
  digestAlgorithms  DigestAlgorithmIdentifiers,
  contentInfo       ContentInfo,
  certificates      [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
  crls              [1] IMPLICIT CertificateRevocationLists OPTIONAL,
  signerInfos       SignerInfos }
DigestAlgorithmIdentifiers ::=  SET OF DigestAlgorithmIdentifier
SignerInfos ::= SET OF SignerInfo
```

Podepsaná data

Pomocí datového typu ***SignedData*** se ukládá digitální podpis dat – viz rámeček 2 a obrázek 1. Vstupní data jsou uložena v položce *contentInfo* a mohou být podepsána několika subjekty. Dílčí podpisy spolu s dalšími informacemi tvoří dílčí položky *signerInfo*, které souborně za všechny signatáře vytvářejí položku *signerInfos*. Každý signatář může použít jiný podepisovací algoritmus i hašovací funkci a další údaje (viz definice položky *signerInfo* v rámečku 3).

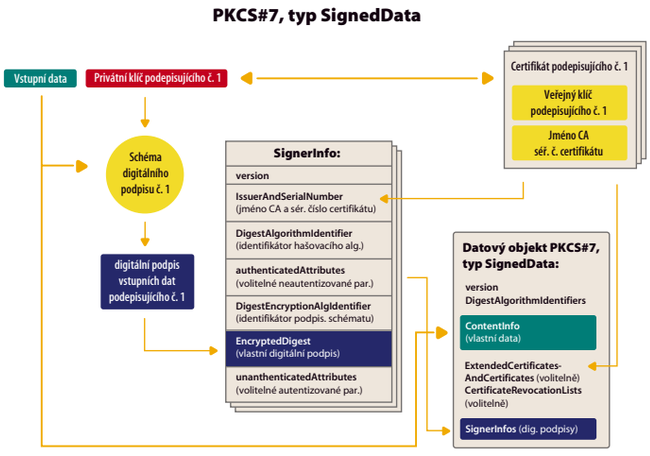
Údaj *version v signerInfo* označuje jako obvykle verzi syntaxe tohoto datového typu a *issuerAndSerialNumber* určuje certifikát signatáře. Ten je uložen až v položce *certificates* a získáváme z něho veřejný klíč při ověřování daného dílčího podpisu. V položce *digestAlgorithm* je určen hašovací algoritmus, kterým signatář hašuje vlastní data a autentizované atributy. Autentizované atributy *(authenticatedAttributes)* jsou další data, která může signatář podepsat spolu s vlastními daty (například čas podpisu), zatímco ty, které nechce autentizovat *(unauthenticatedAttributes)*, jsou uvedeny až za vlastním podpisem.

Identifikace algoritmu, který byl použit pro vytvoření podpisu, je v položce *digestEncryptionAlgorithm* a konečně vlastní podpis dat a autentizovaných parametrů je uložen v položce *encryptedDigest*. Další informace k autentizovaným a neautentizovaným parametrům je možné nalézt v PKCS#9.

‹3› Podpis a informace o signatáři

```
SignerInfo ::= SEQUENCE {
  version                   Version,
  issuerAndSerialNumber     IssuerAndSerialNumber,
  digestAlgorithm           DigestAlgorithmIdentifier,
  authenticatedAttributes   [0] IMPLICIT Attributes OPTIONAL,
  digestEncryptionAlgorithm DigestEncryptionAlgorithmIdentifier,
  encryptedDigest           EncryptedDigest,
  unauthenticatedAttributes [1] IMPLICIT Attributes OPTIONAL }

EncryptedDigest ::= OCTET STRING
```



Obr. 1. PKCS#7, typ „Podepsaná data“

Obalená data

Tzv. digitální obálka dat je sofistikovaný datový typ ***EnvelopedData***, který je definován v rámečku 4. Je určen k přenosu zašifrovaných dat *(EncryptedContentInfo)* libovolnému počtu příjemců. Data jsou šifrována symetrickým šifrovacím algoritmem, jehož identifikátor je uložen v položce *contentEncryptionAlgorithm* a jehož klíč je vygenerován náhodně. Klíč označme například DEK (data encrypting key). DEK je pak uložen pro každého příjemce do jemu odpovídající „hlavičky" *(RecipientInfo)* a všechny tyto hlavičky jsou pak uloženy do položky *RecipientInfos* v *EnvelopedData*.

Zbývá vysvětlit, jak je DEK pro daného příjemce uložen v jeho hlavičce *RecipientInfo*. Z rámečku 4 a obrázku 2 je vidět, že princip je velmi jednoduchý. DEK je prostě jen zašifrován veřejným klíčem tohoto příjemce a takto uložen v *RecipientInfo* jako položka *EncryptedKey*. Veřejný klíč příjemce se zjistí pomocí nám už známé položky *issuerAndSerialNumber* a použitou asymetrickou šifru identifikujeme pomocí *keyEncryptionAlgorithm* (obě položky jsou v *RecipientInfo* před zašifrovaným klíčem). Připomeňme, že obsahem šifrovaných dat může být jakákoliv forma *ContentInfo* (třeba desetkrát různě šifrovaný a různě podepsaný obsah).

‹4› Obalená data

```
EnvelopedData ::= SEQUENCE {
  version Version,
  recipientInfos RecipientInfos,
  encryptedContentInfo EncryptedContentInfo }

RecipientInfos ::= SET OF RecipientInfo

RecipientInfo ::= SEQUENCE {
  version                 Version,
  issuerAndSerialNumber   IssuerAndSerialNumber,
  keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier,
  encryptedKey            EncryptedKey }
EncryptedKey ::= OCTET STRING
EncryptedContentInfo ::= SEQUENCE {
  contentType             ContentType,
```

```
  contentEncryptionAlgorithm   ContentEncryptionAlgorithmIdentifier,
  encryptedContent             [0] IMPLICIT EncryptedContent OPTIONAL }
EncryptedContent ::= OCTET STRING
```

Podepsaná a obalená data

Tento datový typ, ***SignedAndEnvelopedData***, je typicky určen k přenosu zašifrovaných a současně podepsaných dat několika příjemcům, přičemž podpis může učinit také několik signatářů. K zašifrování dat se používá stejná metoda jako v předchozím případě. Navíc se do této struktury ukládá jen podpis (jednoho nebo více signatářů, což je v zásadě jedno). Zajímavé je pouze to, že i tento podpis se nakonec šifruje stejným klíčem DEK, jako jsou šifrována samotná přenášená data. Ostatní detaily vyplývají ze syntaxe, která je vidět v rámečku 5.

Pro představu si popíšeme postup, jakým se příjemce takové zprávy dostane k datům a jak ověří jejich podpis. Příjemce si nejprve z položky *recipientInfos* (rámeček 4) vybere svoji položku recipientInfo a odtud s použitím svého privátního asymetrického klíče odšifruje uložený klíč DEK. Tímto klíčem a symetrickým algoritmem, jehož identifikátor je *contentEncryptionAlgorithm* v položce *encryptedContentInfo*, odšifruje data uložená v *encryptedContentInfo* a stejným způsobem odšifruje i obsah položky *encryptedDigest* (v *signerInfo* je šifrovaná jen tato položka). Získá tak v prvním případě otevřený tvar dat a v druhém případě otevřený tvar jejich digitálního podpisu. Jeho platnost pak ověří pomocí signatářova veřejného klíče uloženého v *signerInfo*. Jednoduché, že?

‹5› Obalená a podepsaná data

```
SignedAndEnvelopedData ::= SEQUENCE {
  version                  Version,
  recipientInfos           RecipientInfos,
  digestAlgorithms         DigestAlgorithmIdentifiers,
  encryptedContentInfo     EncryptedContentInfo,
  certificates  [0] IMPLICIT ExtendedCertificatesAndCertificates OPTIONAL,
  crls          [1] IMPLICIT CertificateRevocationLists OPTIONAL,
  signerInfos              SignerInfos }
```

Zašifrovaná data

Tento typ obsahuje jen verzi a položku *encryptedContentInfo*, ve které jsou uložena vlastní šifrovaná data: ***EncryptedData*** ::= SEQUENCE {version Version, encryptedContentInfo EncryptedContentInfo}.

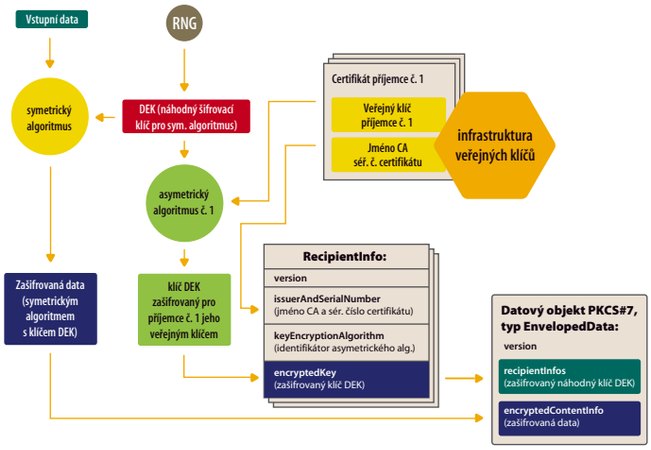
Hašovaná data

Zbývá nám poslední datový typ, ***DigestedData***. Je zaveden zejména pro účely kontroly integrity, jak napovídá jeho definice v rámečku 6. Obsahuje vlastní data v položce *contentInfo* a jejich výtah (digest, haš) v položce *digest*. Použitý hašovací algoritmus identifikuje položka *digestAlgorithm*.

Další vývoj standardu

Přestože standard je velmi flexibilní, časem se ukázaly některé cesty jako slepé a naopak se objevily jiné perspektivní. Proto byly v bulletinu RSA popsány zamýšlené změny, které by se měly objevit ve verzi 1.6 a 2.0, zatím však nedostaly oficiální formu. Verze 1.6 reflektuje zejména podporu platebního protokolu SET (Secure Electronic Transactions), odklon od podpory PEM a zapovězení proprietárního formátu PKCS#6 pro *extensions* (místo toho se podporují rozšíření podle X.509 verze 3).

PKCS#7, typ EnvelopedData



Obr. 2. PKCS#7, typ „Obalená data“

Verze 2.0 přinese vylepšení formátu pro využití různých kryptografických algoritmů (určité problémy nastaly u realizace podepisovacího schématu DSS) a flexibilnější identifikaci klíčů. Tím je míněno, že veřejný klíč nemusí být identifikovatelný pouze na základě jména vydavatele a čísla certifikátu (jak jsme to viděli výše), ale i na základě informace přímo z položky „key owner name“ (jméno majitele klíče). Podstatnou změnou bude i rozsáhlejší podpora symetrické kryptografie, možnost oddělení klíčového hospodářství od zprávy (dnes se nesou všechny informace o klíčích přímo ve zprávě) a možnost využívání klíčového hospodářství, vystavěného také na bázi čistě symetrických algoritmů.

‹6› Hašovaná data

```
DigestedData ::= SEQUENCE {
  version              Version,
  digestAlgorithm      DigestAlgorithmIdentifier,
  contentInfo          ContentInfo,
  digest               Digest }
Digest ::= OCTET STRING
```

Závěr

Seznámili jsme se se standardem PKCS#7, který se stal dominujícím v oblasti šifrování a podepisování elektronické pošty, protože je využíván protokolem S/MIME. Další vývoj standardu bude směřovat mnohem dál. Je zde velká šance, že se stane univerzálním formátem pro zprávy využívající nejrůznější kryptografické techniky, včetně asymetrického a symetrického klíčového hospodářství. Vzniká tak zajímavá situace, kdy formáty pro kryptografii s veřejným klíčem (PKCS) se budou ve své nejpokročilejší verzi stále více obracet k symetrické kryptografii, o níž si mnozí myslí, že ji kryptografie s veřejným klíčem měla zcela nahradit. Pochopitelně je to omyl, obě dvě větve žijí ve velmi dobré symbióze. ❙❙❙

Vlastimil Klíma ❙ *v.klima@decros.cz*