

Jak popsat data

V našem seriálu tentokrát maličko odbočíme. Abychom mohli dále vysvětlovat normy PKCS, musíme se totiž nejprve seznámit s ASN.1. Je to formální jazyk (uznaný i jako česká norma), který umí popsat libovolné abstraktní datové struktury, je platformově nezávislý a umožňuje tak správnou interpretaci dat a jejich výměnu mezi různými platformami a vrstvami architektury OSI.

ASN.1 se používá v telekomunikacích a výpočetní technice. Neobejdeme se bez něj, pokud budeme chtít porozumět klíčovému průmyslovým a mezinárodním normám z oblasti bezpečnosti, například těm pro elektronický podpis, certifikáty a dalším. Tyto normy totiž zavádějí datové struktury, které pak využívá řada jiných norem. „Abstraktní syntaktická notace jedna“ je tedy jazyk umožňující definování abstraktních objektů, jež jsou přenášeny nebo využívány jednotlivými vrstvami architektury OSI. Proto jej kromě PKCS využívá i řada mezinárodních norem (ANSI, ISO, ITU, ITU-T, NIST, W3C, IEC aj.).

Téma ASN.1 je rozsáhlé – jen dvě základní české normy k ASN.1 mají přes sto stran textu. Proto zde vybereme pouze několik pravidel a datových typů a na nich ilustrujeme, co to ASN.1 vlastně je. Nejspíš i pak občas narazíte na zápis, kterému neporozumíte – chce to však jen trpělivost (říkal jsem si to také asi milionkrát...), protože jde o skutečně abstraktní jazyk.

Zájemcům o hlubší studium doporučuji monografii o ASN.1, která právě teď vyšla, a samozřejmě základní normy k ASN.1 (viz infotypy).

<1> Nový datový typ RSAPublicKey

```
RSAPublicKey ::= SEQUENCE {
    modulus      INTEGER, -- číslo n = p*q
    publicExponent INTEGER -- číslo e
}
```

SKORO PROGRAMOVACÍ JAZYK...

Rámeček 1 ukazuje, jak se podle PKCS#1 pomocí ASN.1 definuje formát veřejného klíče. Jak vidíte, syntaxe ASN.1 je intuitivní a podobná programovacím jazykům. Poznamenejme jen, že text za pomlčkami je *komentář*, jména napsaná vesměs velkými písmeny označují *datové typy* tzv. *univerzální třídy* (protože je definuje přímo norma ASN.1), zatímco jména pouze začínající velkým písmenem označují *ostatní datové typy*, a malým písmenem začínají jména *hodnot datových typů*.

Z jednoduchých typů lze skládat složitější typy (*konstruované*), jako je například typ **RSAPublicKey** z rámečku 1. V ASN.1 se vše točí kolem datových typů a hodnot (typ = množina přípustných hodnot, hodnota = prvek množiny daného typu). K jednoznačné identifikaci typu slouží *označení* typu (tag). Typy se dělí do čtyř tříd (viz rámeček 2) a v každé třídě jsou typy očíslovány. Proto se uvádějí ve tvaru [*třída číslo*]. Například INTEGER je *jednoduchý datový typ*, který vyjadřuje celá čísla a má označení [UNIVERSAL 2]. Nové typy i hodnoty typů mohou být libovolně pojmenovány pomocí operátoru přiřazení ::= a tato jména lze dále používat pro definici dalších typů a hodnot.

<2> Třídy typů

V ASN.1 rozeznáváme čtyři třídy:

Universal – pro typy, které jsou definovány přímo normou ASN.1 (viz tabulka). Tyto typy mohou být používány ve všech normách a aplikacích.

Application – pro datové typy, jejichž význam je specifický pro určitou aplikaci, například typy z normy X.500 (adresářové služby).

Private – pro typy, jejichž význam je specifický v rámci nějaké firmy.

Context-specific – pro typy, jejichž význam je specifický pro daný konstruovaný typ; slouží k odlišení komponent strukturovaných typů.

TRIK S OBJEKTIVÝM IDENTIFIKÁTOREM

Všechny jednoduché typy mají mnemotechnická jména, která vidíte v tabulce. Například BIT STRING je libovolný řetězec bitů, IA5 STRING je libovolný řetězec znaků mezinárodní abecedy číslo 5 (ASCII), INTEGER je libovolné celé číslo, NULL je prázdná hodnota, OCTET STRING je libovolný

řetězec oktětů (osmice bitů) atd. Univerzální typ OBJECT IDENTIFIER je tzv. objektový identifikátor (číslo). Využívá se k přehledné identifikaci za ním následujících datových typů z jiných norem, které je potřeba využít v definici nového typu. Jeho hodnotou je posloupnost komponent, například {1 2 840 113549 1 9 1}, které identifikují daný objekt dané normy (v tomto případě je to objekt dané normy (v tomto případě je to objektový identifikátor pro typ „emailAddress“). Pokud tedy „obyčejné“ hodnotě **v.klima@decros.cz** typu IA5 STRING předřadíme uvedený objektový identifikátor (viz rámeček 6), je nyní tento řetězec chápán již jako e-mailová adresa. Je to tedy jakýsi příznak, který umožňuje elegantně interpretovat a v podstatě i rozšířit datové typy.

Kódování identifikátorů je kapitola sama pro sebe; zde jen pro ilustraci uvedme, že kompo-

číslo typu	označení typu (hex.)	typ
1	0x01	BOOLEAN
2	0x02	INTEGER
3	0x03	BIT STRING
4	0x04	OCTET STRING
5	0x05	NULL
6	0x06	OBJECT IDENTIFIER
7	0x07	OBJECT DESCRIPTOR
8	0x08	EXTERNAL
9	0x09	REAL
10	0x0A	ENUMERATED
11	0x0B	rezervováno
12	0x0C	rezervováno
13	0x0D	EMBEDDED
14	0x0E	rezervováno
15	0x0F	rezervováno
16	0x10	SEQUENCE a SEQUENCE OF
17	0x11	SET a SET OF
18	0x12	NUMERIC STRING
19	0x13	PRINTABLE STRING
20	0x14	TELETEXT STRING
21	0x15	VIDEO STRING
22	0x16	IA5 STRING (ASCII)
23	0x17	UCT TIME
24	0x18	GENERALIZED TIME
25	0x19	GRAPHICAL STRING
26	0x1A	VISIBLE STRING
27	0x1B	GENERAL STRING
28	0x1C	UNIVERSAL STRING
29		rezervováno
30	0x1E	BASIC MULTILINGUAL PLANE STRING

Univerzální typy v ASN.1



nenty {1 2 840 113549 1 9 1} vyjadřují strom „iso (1) – členské země (2) – usa (840) – rsadsi (113549) – skupina norem PKCS (1) – norma PKCS#9 (9) – emailAddress (1)“. Objektové identifikátory mohou pochopitelně vydávat jen registrované organizace.

KONSTRUOVANÉ TYPY

Základními konstruovanými typy ASN.1 jsou: SEQUENCE – uspořádaný seznam jednoho nebo více typů, SEQUENCE OF – uspořádaný seznam žádného nebo více výskytů jednoho daného typu a SET (resp. SET OF) – pro neuspořádaný případ. **Obsah konstruovaného typu vytváříme tak, že jeho položky prostě jen řadíme za sebou** (například modulus, publicExponent u **RSAPublicKey** z rámečku 1). Konstruované typy jsou základem ASN.1, neboť jakýkoliv vzniklý typ můžeme okamžitě použít ke konstrukci nového typu. Tímto způsobem jsou v různých normách definovány stovky nových typů. Zároveň se k nim přiřadí objektové identifikátory a pomocí nich je mohou využívat i další normy (například využití typu z X.509 v PKCS, rámeček 3).

DALŠÍ DRUHY TYPŮ

ASN.1 rozeznává čtyři druhy typů: *jednoduché* (atomické, nemají komponenty), *konstruované* (mají komponenty), *označené typy* (mohou být odvozeny z jakýchkoliv jiných typů) a (neoznačené, variantní) typy CHOICE a ANY. V ASN.1 mají všechny typy označení (jsou tedy *označené*) kromě zmíněných dvou; ty nemohou mít vlastní označení, protože to jsou zá-stupná slova – CHOICE jen přebírá typ z vybrané položky a ANY představuje „nějaký“ typ (předpokládá se, že v době přenosu dat už bude k dispozici konkrétní naplnění typu), viz rámeček 3.

<3> CHOICE a ANY

Pomocí slov CHOICE a ANY přesouváme určení typu na následné položky.

Například typ *ExtendedCertificateOrCertificate* z PKCS#7 je definován takto:

```
ExtendedCertificateOrCertificate ::= CHOICE {
    certificate Certificate,
    extendedCertificate [0] IMPLICIT
        ExtendedCertificate
}
```

Je-li vybrána první alternativa, výsledkem je typ Certificate, je-li vybrána druhá alternativa,

typ výsledku je [0] IMPLICIT ExtendedCertificate. V druhém případě se jedná o tzv. implicitní označování.

ANY se může použít samostatně: např. `NášTyp ::= ANY`, pak je typ neurčitý, ale častěji se používá s „ANY DEFINED BY“. Často využívaný typ *AlgorithmIdentifier* z normy X.509 je definován jako:

```
AlgorithmIdentifier ::= SEQUENCE {
    algorithm OBJECT IDENTIFIER,
    parameters ANY DEFINED BY algorithm
        OPTIONAL
}
```

Tato konstrukce je velmi šikovná, protože zahrnuje i algoritmy, které v dané době nejsou známé.

EXPLICITNÍ A IMPLICITNÍ OZNAČOVÁNÍ

Univerzální typy jsou definovány přímo v ASN.1 včetně označení. Ostatní typy mohou být definovány jinde (viz rámeček 2) a jejich *označení* je pak vždy dáno buď implicitně, nebo explicitně. V prvním případě používáme klíčové slovo IMPLICIT; typ, který je takto definován, přebírá

označení typu následujícího za slovem IMPLICIT (v rámu 3 je to **ExtendedCertificate**).

EXPLICIT se používá, chceme-li se vyvarovat možného nedorozumění v označení typu, a proto typ přeznačíme (viz rámeček 4). Původnímu typu totiž můžeme pomoci EXPLICIT ještě *předřadit* vnější označení, čímž ho odlišíme od ostatních typů.

```
<4> Notace ASN.1 pro explicitní označování
[[class] number] EXPLICIT Type
class = UNIVERSAL | APPLICATION |
PRIVATE
```

Například typ ContentInfo z PKCS#7 je definován jako

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

Druhé komponentě se předřazuje vnější označení – má číslo nula (v hranaté závorce) a třídu context-specific (v případě, že *class* chybí, jde o třídu context-specific). Navíc jde o nepovinnou komponentu (OPTIONAL).

KÓDOVACÍ PRAVIDLA

Teď už jsme jakžtakž zvládli něco z jazyka ASN.1, ale k přenosu dat to nestačí. Báječně nadefinované abstraktní datové hodnoty je ještě potřeba konkrétně zakódovat do posloupnosti nul a jedniček. ASN.1 má

infotypy

Česká norma k ASN.1 – ČSN ISO/IEC 8824 přebírá text mezinárodní normy 8824:

X.680: ITU-T Recommendation X.680 (1997). **ISO/IEC 8824-1:1998.** *Information Technology – Abstract Syntax Notation One (ASN.1): Specification of Basic Notation*

Česká norma ke kódování BER – ČSN ISO/IEC 8825 přebírá text mezinárodní normy 8825:

X.691: ITU-T Recommendation X.691 (1997). **ISO/IEC 8825-2:1998.** *Information Technology – ASN.1 Encoding Rules: Specification of Packed Encoding Rules (PER)*

Kódování:

X.690: ITU-T Recommendation X.690 (1997). **ISO/IEC 8825-1:1998.** *Information Technology – ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

Monografie o ASN.1:

Larmouth J.: *ASN.1 Complete*. Morgan Kaufman Publ., 2000

možnost volby mezi několika typy kódování (BER, CER, DER a PER – viz infotypy). Táž data lze tedy zakódovat různým způsobem vhodným pro daný přenos (interpretace dat ale zůstává vždy stejná).

Nejčastějším kódováním je BER (a bývá často automaticky spojováno s ASN.1). Umožňuje zakódovat hodnotu více způsoby, např. řetězec „abc“ v ASCII lze v BER zakódovat pomocí jednoduchého typu IA5 STRING nebo pomocí konstruovaného typu (z IA5 STRING), viz rámeček 5. DER je zúžením pravidel BER tak, aby existovala pouze jediná cesta, jak data zakódovat a rozkódovat. CER je další zvláštní forma BER určená pro kódování zpráv, u nichž v době kódování ještě neznáme jejich celkovou délku. PER je poměrně nový typ kódování používaný např. v řízení letového provozu nebo audiovizuálních přenosech.

<5>Kódování BER a DER

Příklad zakódování hodnoty „test“ typu IA5 STRING (0x16):

DER: 16 04 74 65 73 74

BER (dlouhá forma délkových oktětů):

16 81 04 74 65 73 74

BER – konstruovaný typ. tvar „tes“ + „t“:

36 07 16 03 74 65 73 16 01 74

PRINCIPY KÓDOVÁNÍ

Každou datovou hodnotu v kódování BER kódujeme ve čtyřech částech, které následují za sebou. Jsou to: identifikátorové oktety, délkové oktety, obsahové oktety a oktety konce obsahu. V základním kódování, kdy je délka dat známa, se čtvrtá část nepoužije a vše je jednoduché: identifikátorové oktety říkají, jaký typ dat máme očekávat, délkové určují počet obsahových oktětů a v obsahových jsou vlastní data.

IDENTIFIKÁTOROVÉ OKTETY

Tyto oktety určují třídu typu, příznak (jednoduchý/konstruovaný) a číslo typu v rámci dané třídy. Je-li číslo typu menší než 31 (tzv. *krátký tvar označení*), identifikátorové oktety se skládají z jednoho oktetu, je-li větší nebo rovno 31 (*dlouhý tvar označení*), skládají se z více oktětů.

V prvním případě mají bity identifikátorového oktetu následující význam: bity 8 a 7 kódují třídu datového typu (Universal = 00, Application = 01, Context Specific = 10, Private = 11), bit 6 kóduje jednoduchý (0) nebo konstruovaný (1) typ a zbylé bity kódují číslo typu v rámci jeho třídy.

V druhém případě jsou bity 8 až 6 prvního oktetu nastaveny stejně jako dříve, ale v bitech 5 až 1 jsou jedničky. V dalších oktetech je bit 8 nulový, vyjma posledního, který zde má 1. Škrtneme-li z těchto oktětů bity 8 jako nevýznamné, zůstanou ve zbylých bitech požadované číslo v bázi 128.

DÉLKOVÉ OKTETY

Opět jsou zde dva případy – délka do 127 oktětů a větší. V prvním případě (krátká forma) je délkový oktet jeden, bit 8 má nulový a ve zbytku je zakódována délka. Dlouhá forma může mít 2 až 127 délkových oktětů. Bit 8 prvního oktetu má hodnotu 1 a zbylé bity udávají počet přídatných oktětů; v těch je pak v bázi 256 vyjádřena délka.

OBSAHOVÉ OKTETY

Jak vypadají obsahové oktety, asi bez dlouhého výkladu nejlépe ilustruje rámeček 6.

<6> Dekódování ASN.1

Následující data, která vidíte v binární formě na obrázku, jsou součástí autorova certifikátu (podle normy X.509) a vznikla zakódováním datové hodnoty typu Name (tento typ je definován v normě X.501).

První oktet udává, že celá vybraná datová struktura je typu SEQUENCE (z jeho hodnoty 0x30 zjistíme, že jde o konstruovaný typ).

Délka dat daná druhým oktetem 0x4F je $4 \cdot 16 + 15 = 79$ oktětů. V obsahových oktetech očekáváme jednotlivé položky struktury. První z nich je SET (identifikátorový oktet je 0x31) o délce 0x0B = 11 oktětů. Protože je to opět konstruovaný typ, hledáme jeho dílčí položky – první z nich je opět SEQUENCE (0x30) s délkou dat 9 (0x09).

Její položkou je objektový identifikátor (0x06) s délkou 3 (0x03) a hodnotou 0x55 0x04 0x06, což po dekódování znamená identifikátor countryName (2 5 4 6).

Další položkou je PrintableString (0x13), má 2 datové oktety (0x02) a hodnotu „CZ“ (0x43 0x5A).

Sečteme-li všechny oktety, vidíme, že zde končí naposledy otevřená struktura SEQUENCE i před ní otevřená struktura SET. Vracíme se tedy k první otevřené struktuře SEQUENCE – další data kódují její druhou položku. Takto bychom dekodovali zbylé data:

```
SEQUENCE {
  SET {
    SEQUENCE {
      OBJECT IDENTIFIER countryName (2 5 4 6)
```

```

PrintableString 'CZ'
}
}
SET {
  SEQUENCE {
    OBJECT IDENTIFIER commonName (2 5 4 3)
    PrintableString 'RNDr. Vlastimil KLIMA'
  }
}
SET {
  SEQUENCE {
    OBJECT IDENTIFIER emailAddress
      (1 2 840 113549 1 9 1)
    IA5 STRING 'v.klima@decros.cz'
  }
}
}
}

```

Pro ilustraci – datový typ Name je v X.501 definován v notaci ASN.1 postupně takto:

```

Name ::= CHOICE { RDNSSequence }
RDNSSequence ::= SEQUENCE OF
  RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF
  AttributeValueAssertion

```

Offset:	Bytes:	ANSI Text:
00000090	31 2D 30 2B 06 03 55 04 03 13 24 4B 50 4E 51 77	1-0+00U000\$KPNQw
000000A0	65 73 74 20 43 7A 65 63 68 69 61 20 50 75 62 6C	est Czechia Publ
000000B0	69 63 20 54 65 73 74 20 43 41 20 32 30 30 30 30	ic Test CA 20000
000000C0	1E 17 0D 30 30 31 30 32 33 30 38 30 39 35 37 5A	000001023080957Z
000000D0	17 0D 30 30 31 31 32 32 30 38 30 39 35 36 5A 30	00001122080956Z0
000000E0	4F 31 0B 30 09 06 03 55 04 06 13 02 43 5A 31 1E	0100000U0000CZ10
000000F0	30 1C 06 03 55 04 03 13 15 52 4E 44 72 2E 20 56	0000U0000RNDr. V
00000100	6C 61 73 74 69 6D 69 6C 20 4B 4C 49 4D 41 31 20	lastimil KLIMA1
00000110	30 1E 06 09 2A 86 48 86 F7 0D 01 09 01 16 11 76	0000 *+H+÷00000v
00000120	2E 6B 6C 69 6D 61 40 64 65 63 72 6F 73 2E 63 7A	.klima@decros.cz
00000130	30 81 9F 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01	00Y0000 *+H+÷0000
00000140	05 00 03 81 8D 00 30 81 89 02 81 81 00 EB 86 D3	0 000 00%000 ětÓ

Pohled na binární vyjádření analyzované části certifikátu (v ASN.1) podle normy X.509

```

AttributeValueAssertion ::= SEQUENCE {
  AttributeType,
  AttributeValue
}
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY

```

SHRNUTÍ

ASN.1 je abstraktní jazyk, který má svoji abecedu, slova a pravidla; jeho prostřednictvím lze popisovat různé abstraktní datové typy a jejich hodnoty a tyto typy také přebírat z jiných norem. Takto vyjádřená data, následně zakódovaná pomocí BER nebo DER, už mohou spolehlivě cestovat různými vrstvami architektury OSI.

III Vlastimil Klíma (v.klima@decros.cz)