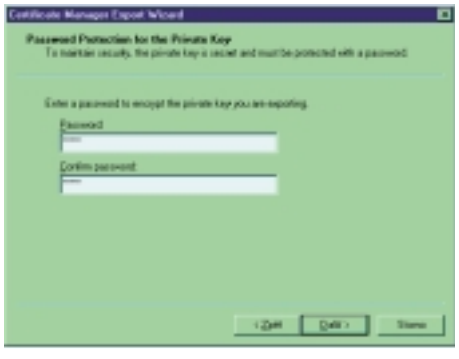
PKCS#12

Důvěrnosti podle normy

Až se objeví první systémy realizující elektronický podpis podle zákona č. 227/2000 Sb., přirozeným cílem hackerů budou také nejcitlivější data, třeba i privátní podepisovací klíče občanů. Povšimněme si proto standardu PKCS#12, který se používá k ochraně těchto klíčů při jejich vzniku, exportu a importu v rámci různých systémů.

Standard PKCS#12 je součástí rodiny standardů PKCS (Public Key Cryptography Standards), se kterou jsme vás seznámili již v Chipu 8/00 (viz infotipy). Tyto standardy se týkají zejména kryptografie s veřejným klíčem, ale také symetrických šifer, hašovacích funkcí, zpracování passwordů apod. PKCS#12 je typickým příkladem skloubení řady těchto technik a má na starosti syntaxi pro výměnu personálních informací (Personal Information Exchange Syntax). Týká se takových osobních identifikačních informací, jakými jsou například privátní asymetrické klíče, certifikáty, seznamy zneplatněných certifikátů a rozličná tajná data. Různé stroje, informační kiosky a aplikace mohou tento standard použít k importu a exportu uvedených typů informací. PKCS#12 je podporován v softwaru Microsoftu (například v Internet Exploreru), Netscapu (Netscape Communicator), v komerčních verzích PGP a v mnoha dalších programech a systémech.

My se zde soustředíme na informace nejcitlivější, na privátní asymetrické klíče. Slouží například k vytváření zaručeného digitálního podpisu a náš zákon o elektronickém podpisu je nazývá „data pro vytváření podpisu“.



Při exportu je privátní klíč chráněn tajným heslem.

PDU A PFX

Při předávání dat podle PKCS#12 hraje důležitou roli tzv. *jednotka PDU (Protocol Data Unit)*, která představuje základní strojově nezávislý formát zprávy v protokolu výměny dat. *PFX (Private File Exchange)* je pak nejvyšší jednotkou, která vstupuje a vystupuje z PKCS#12 (často se setkáte se souborem s koncovkou *pfx* při exportu nebo importu privátního klíče například v Internet Exploreru). Formát PFX kromě **utajení** řeší i velmi důležitou otázku **integrity** předávaných informací. V obou případech jsou k dispozici dvě metody – máme tedy dva mody utajení a dva mody integrity, celkem tedy čtyři kombinace ochrany.

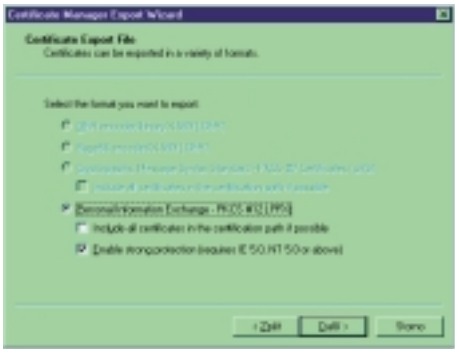
MODY UTAJENÍ

Utajení se provádí buď asymetrickým, nebo symetrickým způsobem. V prvním případě *(Public Key privacy mode)* je informace zašifrována příjemcovým veřejným klíčem. Jde o typ „obalená data“, který jsme definovali v článku o PKCS#7 v minulém čísle (viz infotipy). V druhém případě se data zašifrují symetricky, a to zvoleným algoritmem a klíčem, který je odvozen z uživatelského jména a osobního přístupového hesla (passwordu). Technika odvození je definována normou PKCS#5.

PKCS#12 zahrnuje řadu kryptografických technik a potvrzuje i úspěšnost symbiózy symetrické a asymetrické kryptografie.

MODY INTEGRITY

Opět lze použít asymetrickou nebo symetrickou techniku. V prvním případě je integrita zajištěna digitálním podpisem, a jde tedy o formát „podepsaná data“, kterým jsme se rovněž zabývali v minulém dílu. K podepsání je pochopitelně použit signatářův privátní asymetrický klíč. →



Export privátního asymetrického klíče z MS Internet Exploreru ve formátu PKCS#12

V druhém případě je neporušenost dat zajištěna tzv. zabezpečovacím kódem zprávy. Označuje se zpravidla anglickou zkratkou *MAC (Message Authentication Code)* a blíže jsme o něm psali v sérii článků v roce 1993 (jsou k dispozici elektronicky na internetu, viz infotipy). MAC je standardní technika, která se používá velmi dlouho, a je i předmětem norem ISO. K výpočtu MAC se používá vhodný symetrický algoritmus a tajný klíč, který je opět odvozen z tajného integritního passwordu (a navíc z tzv. soli a počtu iterací). Na password pro zajištění integrity a pro utajení (mohou být odlišné) je uživatel dotazován při exportu a importu privátní informace.

AUTENTIZOVANÁ A BEZPEČNÁ DATA

K zajištění utajení a integrity je možné použít jakoukoliv kombinaci uvedených modů. Data, která takto zabezpečíme, se nazývají autentizovaná a bezpečná *(AuthenticatedSafe)* a tvoří základní datovou strukturu ve formátu PFX. Poznamenejme k tomu, že pokud použijeme asymetrické techniky, musíme mít zajištěno, že veřejné klíče, které k tomuto účelu používáme, jsou důvěryhodné. K tomu pochopitelně budeme muset být účastni nějaké formy infrastruktury veřejných klíčů *(PKI, Public Key Infrastructure)*.

DATOVÝ TYP PFX

Datový typ PFX je definován v rámečku 1 a v nejvyšší rovině je to jednoduše posloupnost tří složek: verze formátu, zabezpečené položky *authSafe* a volitelného kódu MAC pro symetrický případ zabezpečení. Položka *authSafe* může být buď typu „Data“ (je-li zabezpečení realizováno pomocí symetrického MAC), nebo „Podepsaná data“ (v případě zabezpečení pomocí asymetrického digitálního podpisu). Tím je vyřešena základní konstrukce.

Vlastní zabezpečené položky jsou uloženy uvnitř pole **content** položky „Data“ nebo „Podepsaná data“ (tyto datové typy jsou definovány v PKCS#7 a obsahují dvě položky – typ obsahu *contentType* a vlastní obsah *content*). Zabezpečené položky jsou pak konkrétně do pole **content** vnořeny jako zakódované hodnoty (BER) typu **AuthenticatedSafe** (viz rámeček 1) – tento typ už může být košatý, jak potřebujeme. Je to totiž posloupnost obecných kryptografických struktur **ContentInfo** podle PKCS#7, jak ukazuje rámeček 1.

Pochopitelně nám půjde především o zašifrované položky, proto zde uložené datové typy budou zejména „zašifrovaná data“ (encryptedData) v symetrickém případě nebo „obalená data“ (envelopedData) v asymetrickém případě. V každé z položek *content* budou pak už uložena inkriminovaná data (jsou to jednotlivé instance **SafeContents**), a to v kódování BER. **SafeContents** jsou složeny ještě z menších jednotek, a to jako posloupnost položek **SafeBag**, jak ukazuje rámeček 2. Tato konstrukce se zdá opravdu příliš obecná, ale na druhou stranu umožňuje zabezpečení a uložení všech potřebných typů informací.

BEZPEČNÝ KUFŘÍK

Každá položka **SafeBag** (bezpečný kufřík) obsahuje jednu atomární privátní informaci. Prozatím je pro ni definováno pět základních typů: *KeyBag, PKCS8ShroudedKeyBag, CertBag, CRLBag* a *SecretBag*. První dva typy jsou určeny pro uložení privátních klíčů, a to buď v otevřené podobě (pak je to KeyBag ::= PrivateKeyInfo), nebo v zašifrované podobě (v tomto případě je to PKCS8ShroudedKeyBag ::= EncryptedPrivateKeyInfo). Nové typy *PrivateKeyInfo* a *EncryptedPrivateKeyInfo* jsou definovány v PKCS#8, ale můžete je vidět i v rámečku 3. Další dva typy **SafeBag** se týkají certifikátů a seznamu zneplatněných certifikátů. Jejich definice vychází ze struktur normy X.509, se kterou jsme vás seznámili v lednovém Chipu (viz infotipy). Poslední základní typ *SecretBag* je určen pro uložení jakýchkoliv dalších tajných hodnot, přičemž jejich identifikátory jsou otevřené pro další specifikace.

Aby bylo umožněno jakékoliv rekurzivní vnořování **SafeBag** do sebe, jako šestý typ je možné použít celou strukturu **SafeContents**. Další variabilitu přináší i samotná definice **SafeBag** – jak plyne z definice, **SafeBag** obsahuje identifi-

<1> Základní datové typy PKCS#12

```
PFX ::= SEQUENCE {
    version     INTEGER {v3(3)}(v3,...),
    authSafe    ContentInfo,
    macData     MacData OPTIONAL
}
MacData ::= SEQUENCE {
    mac         DigestInfo,
    macSalt     OCTET STRING,
    iterations  INTEGER DEFAULT 1
}

AuthenticatedSafe ::= SEQUENCE OF ContentInfo
ContentInfo obsahuje:
    – Data, jestliže nejsou šifrovaná
    – EncryptedData, jestliže jsou symetricky šifrovaná
    – EnvelopedData, jestliže jsou asymetricky šifrovaná

ContentInfo ::= SEQUENCE {
    contentType   ContentType,
    content       [0] EXPLICIT ANY DEFINED BY
                      contentType OPTIONAL }
kde
ContentType ::= { data | signedData | envelopedData | signedAndEnvelopedData | digestedData | encryptedData }
```

<2> Typy SafeContents a SafeBag

```
SafeContents ::= SEQUENCE OF SafeBag

SafeBag ::= SEQUENCE {
    bagId          BAG-TYPE.&id ({PKCS12BagSet})
    bagValue       [0] EXPLICIT BAG-TYPE.&Type
                   ({PKCS12BagSet}{@bagId}),
    bagAttributes  SET OF PKCS12Attribute
                   OPTIONAL}

PKCS12Attribute ::= SEQUENCE {
    attrId      ATTRIBUTE.&id ({PKCS12AttrSet}),
    attrValues  SET OF ATTRIBUTE.&Type
                ({PKCS12AttrSet}{@attrId})}

PKCS12AttrSet ATTRIBUTE ::= {
    friendlyName | localKeyId,        – z PKCS #9
    ... – další atributy je možné dodefinovat}

bagtypes OBJECT IDENTIFIER ::= {pkcs-12 10 1}

v PKCS#12 jsou definovány tyto identifikátory typů:
keyBag  BAG-TYPE ::=
    {KeyBag IDENTIFIED BY {bagtypes 1}}
pkcs8ShroudedKeyBag BAG-TYPE ::=
    {PKCS8ShroudedKeyBag IDENTIFIED BY {bagtypes 2}}
certBag BAG-TYPE ::=
    {CertBag IDENTIFIED BY {bagtypes 3}}
crlBag BAG-TYPE ::=
    {CRLBag IDENTIFIED BY {bagtypes 4}}
secretBag BAG-TYPE ::=
    {SecretBag IDENTIFIED BY {bagtypes 5}}
safeContentsBag BAG-TYPE ::=
    {SafeContents IDENTIFIED BY {bagtypes 6}}

PKCS12BagSet BAG-TYPE ::= { keyBag | pkcs8ShroudedKeyBag | certBag | crlBag | secretBag | safeContentsBag, ... – a v budoucnu další }
```

<3> Typy PrivateKeyInfo a EncryptedPrivateKeyInfo

```
PrivateKeyInfo ::= SEQUENCE {
  version             Version,
  privateKeyAlgorithm PrivateKeyAlgorithmIdentifier,
  privateKey          PrivateKey,
  attributes          [0] IMPLICIT Attributes OPTIONAL }

Version ::= INTEGER
PrivateKeyAlgorithmIdentifier ::= AlgorithmIdentifier
PrivateKey ::= OCTET STRING
Attributes ::= SET OF Attribute

EncryptedPrivateKeyInfo ::= SEQUENCE {
  encryptionAlgorithm EncryptionAlgorithmIdentifier,
  encryptedData       EncryptedData }

EncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
EncryptedData ::= OCTET STRING
```

→ kátor typu, potom vlastní hodnotu a nakonec atributy. V atributech je možné ukládat takové informace, jako jsou identifikátory klíčů, přezdívky a další vizuálně zobrazitelné informace, které slouží buď uživatelům k orientaci, nebo k managementu. Tímto posledním atomárním typem je celá struktura PFX kompletně vyčerpána. Nyní se podíváme, jak se struktura PFX zabezpečuje a šifruje.

VÝPOČET MAC

MAC je zvláštní funkce, jejímž vstupem je zabezpečovaná zpráva (M) a tajný integritní klíč (K) a výstupem je zabezpečovací kód MAC. Za svou nepadělatelnost vděčí MAC skutečnosti, že vypočítat ho a zkontrolovat může jen ten, kdo zná jak zprávu M, tak tajný klíč K. Tajný klíč K je však odvozen od tajného integritního passwordu, takže případný útočník je v tomto případě ve zcela jiné situaci, než když zabezpečení zajišťuje například jen známý kód CRC (Cyclic Redundancy Check) – ten lze bez problémů padělat.

V PKCS#12 je použit MAC, který se označuje HMAC a je postaven na využití hašovací funkce (viz infotipy), kterou označíme H. Může jí být jakákoliv hašovací funkce, ale používá se zejména SHA-1; od MD5 už se ustupuje z bezpečnostních důvodů. Výpočet HMAC je přesně definován v RFC 2104 (viz infotipy), ale v zásadě lze říci, že kombinace klíče K s hašovací funkcí H a zprávou M probíhá tak, že se nejprve klíč doplní nulovými bity na délku bloku B bajtů (zde B = 64) a definují se konstantní B-bajtové bloky *ipad* a *opad;* výsledný kód se pak vypočítá jako HMAC = H((K XOR opad) || H((K XOR ipad) || M)), kde || značí zřetězení.

VÝPOČET SYMETRICKÝCH KLÍČŮ

Pro šifrování vlastních dat v jednotlivých podstrukturách PFX se používají symetrické algoritmy. K šifrování máme k dispozici tajný password, na nějž se program uživatele dotazuje při otevírání nebo ukládání souboru typu PFX s privátními informacemi. Z této tajné hodnoty a pomocí vhodného systému, který generuje náhodná data, můžeme derivovat pokaždé jiný klíč a jiný inicializační vektor (IV) pro šifrování. Jak se IV použije, jsme popsali v článku o proudových a blokových šifrách v Chipu 7/00 (viz infotipy).

Vlastní hodnota klíče a inicializačního vektoru se vypočítává poměrně složitým způsobem. Detailní postup je popsán v normě samé a je navíc závislý na dalším parametru, tzv. počtu iterací (r). Místo hašovací funkce H se pak používá $H^r(M) = H(H(H..(H(M))...)$ a parametr r se volí obvykle v řádu tisíců (!). Aby přijímací strana věděla, jaké parametry byly použity, sůl i počet iterací se ukládá do PFX, konkrétně do struktury **pkcs-12PbeParams** ::= SEQUENCE {**salt** OCTET STRING, **iterations** INTEGER}.

Zkratka Pbe znamená *Password based encryption* (šifrování založené na passwordu) a norma definuje objektové identifikátory pro použití Pbe s algoritmy RC2, tripleDES a RC4, a to s různými délkami klíče.

ZÁVĚR

Seznámili jsme se se standardem PKCS#12, který je široce používán pro přenos privátních informací. Mezi jeho nejdůležitější nasazení patří ochrana privátních klíčů; je jistě zřejmé, že zanedbání právě tohoto aspektu se může ošklivě vymstít. Standard v sobě kumuluje mnoho technik, které jsou popsány v ostatních normách rodiny PKCS, a je v něm dobře vidět provázanost symetrické a asymetrické kryptografie i možnosti praktického využití mnoha technik v praxi.

Vlastimil Klíma | v.klima@decros.cz

INFOTIPY

Přehled standardů PKCS:
Návrat šampiona, Chip 8/00, str. 40–42

O PKCS#7:
I šifra musí mít formát, Chip 3/01, str. 137–139

O MAC:
Nepadělatelné zabezpečení dat, Chip 10/93, str. 252 – 253 a Chip 11/93, str. 194–195

O normě X.509:
Kdopak se to podepsal?, Chip 1/01, str. 130–133

Standard PKCS#12
▸ *www.rsasecurity.com/rsalabs/pkcs/*

Hašovací funkce:
Jak se melou data, Chip 4/99, str. 44–46

O HMAC:
RFC 2104: Keyed-Hashing for message Authentication, IETF, 1997

Proudové a blokové šifry:
Šifry s mnoha tvářemi, Chip 7/00, str. 50–53

Články z Chipu jsou též k dispozici elektronicky na adrese
▸ *www.decros.cz/security_division/crypto_research/archiv.htm*