KAPITOLA 7

Vylepšená kryptografie

Přehled

Všechny verze Windows vylepšují kryptografii, ale ve většině případů jde o několik nových rozhraní API či nových algoritmů. Windows Vista jsou odlišné, protože Microsoft přidal novou moderní kryptografickou infrastrukturu, zvanou Cryptography API: Next Generation (CNG), jež obsahuje nová rozhraní API a nabízí podporu jádra i uživatelského režimu, vylepšenou podporu kryptografické pružnosti (crypto-agility), nové šifrovací sestavy [především Suite B (NSA 2005)] a zlepšené audity.

Microsoft také pomocí nových kryptografických algoritmů a podpory Suite B vylepšil vrstvu Secure Sockets Layer a Transport Layer Security (SSL/TLS). V této kapitole si popíšeme všechny tyto prvky i další součásti tématu.

Důležité: Podotýkáme, že tato kapitola nevysvětluje, jak kryptografické algoritmy pracují, a patrně se po jejím přečtení nestanete odborníky na kódování!

Windows Vista podporují následující kryptografická rozhraní v uživatelském režimu:

- CNG.
- Kryptografické API 1.0 (CAPI 1.0).
- Kryptografické API 2.0 (CAPI 2.0).
- Kryptografie v .NET Frameworku.

Kryptografické inovace se objeví v CNG a .NET, přičemž CAPI 1.0 bude nakonec staženo. Podporu bude mít CAPI 2.0, protože nejde o strukturu nadřazenou CAPI 1.0. Můžete namítnout, že název CAPI 2.0 je nešťastný. Je to pravda! CAPI 2.0 je funkčně něco jiného než CAPI 1.0 – CAPI 2.0 slouží ke správě a generování certifikátů X.509 a souvisejících standardů a oproti CAPI 1.0 či CNG není určeno k podpoře základních kryptografických objektů.

Režim jádra a uživatelský režim

Jednou z nejvýznamnějších změn v CNG je zařazení rozhraní API pro uživatelský režim a režim jádra. V předchozích verzích Windows pracovaly technologie jako Cryptographic API (CAPI) pouze v uživatelském režimu a kryptografie v režimu jádra vyžadovala zcela odlišnou množinu API, jako například Microsoft Kernel Mode Cryptographic Module (Microsoft 2000). Jde o velikou výhodu pro vývojáře, kteří vytvářejí kód pro uživatelský režim a režim jádra, protože si nyní musí pamatovat pouze jednu množinu API.

Všimněte si rovněž toho, že CNG má dvě odlišné množiny názvů funkcí. Funkce `NCrypt*` se týkají správy klíčů, životnosti klíčů, izolace klíčů a některých kryptografických operací s veřejnými klíči (protože soukromé klíče nemohou přesáhnout hranice kryptografie – pokud vaše aplikace potřebuje izolaci klíčů). Funkce `BCrypt*` jsou základní kryptografické funkce na nízké úrovni, jež běží v procesu vaší aplikace, a klíče se neukládají, neboť jsou dočasné.

Kryptografická pružnost

Kryptografické algoritmy jsou neustále vystaveny útokům na dvou frontách. První bojovou linií je neustálý nárůst rychlosti procesorů, čímž se zvyšuje šance prolomení algoritmů v rozumných časech, a druhá se týká postupujícího kryptografického výzkumu, který soustavně nachází v algoritmech slabiny. Dobrou ukázkou tohoto výzkumu jsou slabá místa, jež byla nalezena v některých zásadních hašovacích funkcích (hash function) – MD4 a MD5 jsou dnes považovány za velmi zranitelné kvůli malé odolnosti vůči kolizím a SHA-1 rovněž vykazuje známky závažných nedostatků. V návaznosti na tato zjištění zakázal Microsoft používání MD4, MD5 a SHA-1 v novém kódu, s výjimkou zpětné kompatibility či situací, kdy se algoritmus používá v průmyslových standardech nebo v aplikacích, kde minimální vyžadovaná platforma Windows je jakákoliv verze systému starší než Windows Vista.

Další informace: Národní institut pro standardy a technologii (National Institute of Standards and Technology, NIST) nabízí skvělý postup při používání SHA-1: Federální agentury by měly přestat používat SHA-1 pro digitální podpisy, digitální časové značení a další aplikace, které vyžadují kolizní odolnost, jakmile to bude možné, a po roce 2010 přejít na skupinu hašovacích funkcí SHA-2. Po roce 2010 mohou federální agentury používat SHA-1 pouze pro následující aplikace: kódy na autentizaci zpráv pomocí hašování (Hash-based Message Authentication Codes, HMAC), funkce na odvozování klíčů (key derivation functions, KDF) a generátory náhodných čísel (Random Number Generators, RNG). Bez ohledu na typ použití NIST doporučuje návrhářům aplikací a protokolů využívat hašovací funkce SHA-2 pro všechny nové aplikace a protokoly, (NIST 2006).

Potřebu nových algoritmů občas diktuje politika – skvělým příkladem jsou americké federální požadavky „Suite B". Jestliže vaše aplikace nezná Suite B, ale konkurenční produkt ano, poté váš nedostatek v oblasti kryptografické pružnosti může vést ke ztrátě zakázek.

Největší chyba, kterou vývojáři dělají, je zapsání jména algoritmu napevno do kódu; je-li algoritmus slabý a potřebuje nahradit, je nutné pro nový algoritmus kód upravit, datové struktury je potřeba zaktualizovat a na závěr je potřeba novou verzi produktu dopravit k zákazníkům. V některých případech může tento druh činnosti způsobit nekompatibilitu se staršími souborovými formáty. Následující kód, jenž používá kryptografické API (CAPI), je ukázkou kódu bez kryptografické pružnosti, protože je v něm vložen identifikátor algoritmu:

```
if(CryptDeriveKey(hProv, CALG_DES, hHash, CRYPT_EXPORTABLE, &hKey)) {
    // šifrování dat
}
```

Dobrým příkladem kryptografické pružnosti je SSL a TLS, které bezpečně přenášejí kryptografické algoritmy mezi serverem a klientem. Je-li algoritmus shledán slabým, jednoduché nastavení jiného algoritmu na serveru nebo na klientovi zabrání používání tohoto slabého článku. O těchto nastaveních pohovoříme později v této kapitole.

Poznámka: O tom, zdali SSL a TLS skutečně jsou kryptograficky pružné, lze vést spory, poněvadž jejich interní generátory náhodných čísel mají napevno zadané algoritmy SHA-1 a MD5!

Kryptografická pružnost v CNG

V CNG došlo ke třem výrazným vylepšením, která kryptografickou pružnost podporují. V první řadě jsou všechny kryptografické konstanty řetězce, a nikoli číselné konstanty. V CAPI jsou všechny kryptografické algoritmy předdefinovány v souboru *wincrypt.h*. Kvůli tomu je velmi obtížné rozšířit kryptografickou funkcionalitu podle potřeb své aplikace. Přidání vlastního symetrického šifrovacího algoritmu do CAPI – například Serpent (Anderson 1999) – není snadné. Ale toto vše se v CNG změnilo, protože k definici svého algoritmu lze použít libovolnou řetězcovou konstantu, a když se vaše aplikace pokusí použít váš algoritmus, CNG načte pro tento zaregistrovaný název poskytovatele kryptografie.

Kryptografická pružnost CNG sahá za základní kryptografické úkony; je možné vložit vlastní šifrovací moduly pro SSL a TLS nebo zákaznické obálky pro CMS a vlastní certifikační elementy. Dokumentaci s popisem kroků nutných k instalaci a registraci doplňku (add-in) CNG lze najít v SDK pro CNG. Hlavní funkce, která přidává doplněk, se nazývá `BCryptAddContextFunctionProvider`.

```
#define BCRYPT_SERPENT_ALGORITHM L"SERPENT"
status = BCryptAddContextFunctionProvider(
     CRYPT_LOCAL,
     NULL, // výchozí obsah
     BCRYPT_CIPHER_INTERFACE,
     BCRYPT_SERPENT_ALGORITHM,
```

```
        L"Serpent Provider",
        CRYPT_PRIORITY_TOP);
```

Druhým zlepšením je, že CNG narozdíl od CAPI nevyžaduje, aby Microsoft podepsal implementaci. Libovolný tvůrce kryptografie může vytvořit poskytovatele kryptografie CNG. Při implementaci poskytovatele CNG je nutné vložit pouze nezbytnou funkcionalitu. Jestliže například použijete poskytovatele Serpent, nemusí poskytovatel obsahovat podepisovací či hašovací funkce, protože Serpent je pouze symetrický šifrovací algoritmus.

Poslední výhoda spočívá v tom, že aplikace se může v případě potřeby podle určitých kritérií dotazovat CNG na podporované algoritmy. Následující kód ukazuje, jak vypsat všechny základní (primitivní) kryptografické algoritmy:

```
PROVIDER_REFS pProviders = NULL;
DWORD dwBufSize = 0;
const DWORD dwFlags = CRYPT_ALL_FUNCTIONS | CRYPT_ALL_PROVIDERS;
for (DWORD dwInterface = BCRYPT_CIPHER_INTERFACE;
       dwInterface <= BCRYPT_RNG_INTERFACE;
       dwInterface ++) {

     NTSTATUS ret = BCryptResolveProviders(
                           NULL,
                           dwInterface,
                           NULL,
                           NULL,
                           CRYPT_UM,
                           dwFlags,
                           &dwBufSize,
                           &pProviders);
    if (NT_SUCCESS(ret) && pProviders) {
    printf("dwInterface = %d\n", dwInterface);
    for (DWORD k=0; k < pProviders->cProviders; k++) {
        PCRYPT_PROVIDER_REF pProv = pProviders->rgpProviders[k];
        printf("\tFunkce = %S\n", pProv->pszFunction);
        printf("\tPoskytovatel = %S\n", pProv->pszProvider);

    // výpis názvů vlastností
     for ( DWORD j = 0; j < pProv->cProperties; j++)
         printf("\tVlastnost %d = %S\n",
                j,
                pProv->rgpProperties[j]->pszProperty);
     printf("\n");
    }
    BCryptFreeBuffer(pProviders);
    pProviders = NULL;
  }
}
```

Důležité: Poskytovatele CNG smějí instalovat pouze administrátoři.

Nové algoritmy v CNG

CNG nabízí několik novějších algoritmů, z nichž nejznámější a patrně nejdůležitější je podpora Suite B. V tabulkách 7.1 a 7.2 jsou všechny algoritmy, které výchozí poskytovatelé CNG ve Windows Vista podporují.

Tabulka 7.1 Kryptografické algoritmy v CNG ve Windows Vista

Algoritmus	#define	Standard	Povolen v SDL?	Suite B?
RC2	BCRYPT_RC2_ALGORITHM	RFC2288		
RC4	BCRYPT_RC4_ALGORITHM		Ano[*]	
AES	BCRYPT_AES_ALGORITHM	FIPS 197	Ano	Ano
DES	BCRYPT_DES_ALGORITHM	FIPS 46-3, FIPS 81		
DESX	BCRYPT_3DES_ALGORITHM			
3DES	BCRYPT_DESX_ALGORITHM	FIPS 46-3, FIPS 81, SP800-38A		
3DES-112	BCRYPT_3DES_112_ALGORITHM	FIPS 46-3, FIPS 81, SP800-38A		
MD2	BCRYPT_MD2_ALGORITHM	RFC 1319		
MD4	BCRYPT_MD4_ALGORITHM	RFC 1320		
MD5	BCRYPT_MD5_ALGORITHM	FC 132		
SHA-1	BCRYPT_SHA1_ALGORITHM	FIPS 180-2, FIPS 198		
SHA-256	BCRYPT_SHA256_ALGORITHM FIPS	180-2, FIPS 198	Ano	Ano
SHA-384	BCRYPT_SHA384_ALGORITHM	FIPS 180-2, FIPS 198	Ano	Ano
SHA-512	BCRYPT_SHA512_ALGORITHM	FIPS 180-2, FIPS 198	Ano	Ano
RSA (šifrování)	BCRYPT_RSA_ALGORITHM	PKCS#1 v1.5 a v2.0.	Ano	
RSA (podepisování)	BCRYPT_RSA_SIGN_ALGORITHM	PKCS#1 v1.5 a v2.0.	Ano	
Diffie-Hellman	BCRYPT_DH_ALGORITHM	PKCS#3		
Algoritmus digitálního podpisu	BCRYPT_DSA_ALGORITHM	FIPS 186-2		

[*]RC4 je povoleno pouze po úplném kryptografickém přezkoumání.

Tabulka 7.2 Eliptické křivkové kryptografické algoritmy v CNG ve Windows Vista

Algoritmus	#define	Standard
Eliptický křivkový digitální podpis (Elliptic Curve Digital Signature) Algoritmus s křivkou Prime-256 (Algorithm with Prime-256 curve).	BCRYPT_ECDSA_P256_ALGORITHM	FIPS 186-2, X9.62
Eliptický křivkový digitální podpis (Elliptic Curve Digital Signature) Algoritmus s křivkou Prime-384 (Algorithm with Prime-384 curve).	BCRYPT_ECDSA_P384_ALGORITHM	FIPS 186-2, X9.62
Eliptický křivkový digitální podpis (Elliptic Curve Digital Signature) Algoritmus s křivkou Prime-521 (Algorithm with Prime-521 curve).	BCRYPT_ECDSA_P521_ALGORITHM	FIPS 186-2, X9.62
Eliptický křivkový Diffie-Hellman (Elliptic Curve Diffie-Hellman) Algoritmus s křivkou Prime-256 (Algorithm with Prime-256 curve).	BCRYPT_ECDH_P256_ALGORITHM	SP800-56A
Eliptický křivkový Diffie-Hellman (Elliptic Curve Diffie-Hellman) Algoritmus s křivkou Prime-384 (Algorithm with Prime-384 curve).	BCRYPT_ECDH_P384_ALGORITHM	SP800-56A
Eliptický křivkový Diffie-Hellman (Elliptic Curve Diffie-Hellman) Algoritmus s křivkou Prime-521 (Algorithm with Prime-521 curve).	BCRYPT_ECDH_P521_ALGORITHM	SP800-56A

Poznámka: SHA-256, SHA-384 a SHA-512 se souhrnně nazývají SHA-2 a jsou k dispozici ve Windows Vista (v CAPI a CNG), ve Windows Serveru 2003 (v CAPI) a všechny platformy Windows je podporují prostřednictvím .NET Frameworku.

Poznámka: Všechny výše uvedené algoritmy jsou schváleny pro používání v SDL, vyhovují Suite B a jsou v CNG novinkami.

CNG rovněž podporuje dva druhy generátorů náhodných čísel (RNG) a oba tyto generátory jsou povoleny pod SDL: `BCRYPT_RNG_ALGORITHM` a `BCRYPT_RNG_FIPS186_DSA_ALGORITHM`. Většina aplikací používá první z nich, ale pracujete-li s DSA, měli byste použít druhý algoritmus. Oba generátory vyhovují standardům FIPS 186-2 anebo FIPS 140-2.

Práce s CNG

Nyní si představíme některé funkce, které ukazují, jak se pomocí CNG provádějí různé kryptografické úkoly. Berte tyto příklady jako pseudokód s názvy skutečných API. Cílem není demonstrovat každý možný algoritmus nebo kryptografickou operaci ani vás nechceme zahltit obrovskou záplavou kódu. Namísto toho si ukážeme obecný způsob volání API.

Poznámka: Rozhraní API v CAPI1 nemají přístup k poskytovatelům a klíčům CNG, ale CNG umí přistupovat ke klíčům CAPI1, používaným v poskytovatelích kryptografických služeb Microsoft (Microsoft Cryptographic Service Providers).

Windows Vista Software Development Kit obsahuje kompletní ukázky CNG ve složce samples/security/CNG. Pro CNG existujte také samostatné téma v SDK s ukázkami a dokumentací týkající se nastavení CNG a instalace doplňku CNG (Microsoft 2006a).

Ve všech případech je nutné vložit hlavičkový soubor <*bcrypt.h*> a linkovat kód pomocí *bcrypt.dll*. CNG také vrací různé hodnoty stavu definované v souboru ntstatus.h. Také je potřeba přidat do kódu toto makro:

```
#ifndef NT_SUCCESS
#   define NT_SUCCESS(Status) (((NTSTATUS)(Status)) >= 0)
#endif
```

Šifrování dat

```
BCryptOpenAlgorithmProvider(&hAlg,...)
BCryptGetProperty(hAlg,BCRYPT_BLOCK_LENGTH,&dwBlockSize,...)
Alokace zásobníku, zaokrouhlení na následující velikost bloku.
BCryptGetProperty(hAlg,BCRYPT_OBJECT_LENGTH,&cbKeyObjectLen,...)
Alokace zásobníku pro objekt klíče.
BCryptGenerateSymmetricKey(hAlg,&hKey,...)
BCryptEncrypt(hKey,...)
Data jsou nyní šifrována
BCryptDestroyKey(hKey)
BCryptCloseAlgorithmProvider(hAlg,0)
Dealokace zásobníku
```

Všimněte si, že podobně jako `CryptAcquireContext` v CAPI, `BCryptOpenAlgorithmProvider` je celkem náročné volání funkce a může být užitečné cacheovat ve svém kódu návratový ovladač (handle) namísto neustálého otevírání a zavírání poskytovatele.

Hašování dat

```
BCryptOpenAlgorithmProvider(&hAlg,...)
BCryptGetProperty(hAlg,BCRYPT_OBJECT_LENGTH,&cbHash,...)
Alokace zásobníku pro hašování
BCryptCreateHash(hAlg,&hHash,...)
BCryptHashData(hHash,...)
BCryptFinishHash(hHash,...)
Použití hašovaných dat
BCryptDestroyHash(hHash)
BCryptCloseAlgorithmProvider(hAlg,0)
Dealokace zásobníku
```

Kódování MAC

Vytvoření autentizačního kódu zprávy (Message Authentication Code, MAC) je přesně totéž jako vytvoření haše, avšak jsou zde dva rozdíly.

1. Posledním parametrem funkce `BCryptOpenAlgorithmProvider` musí být `BCRYPT_ALG_HANDLE_HMAC_FLAG`.
2. Pátý a šestý parametr funkce `BCryptCreateHash` jsou skrytý klíč MAC a délka tohoto klíče. Volání funkce proto vypadá takto:

```
BCRYPT_ALG_HANDLE hAlg = NULL;
NTSTATUS status = STATUS_UNSUCCESSFUL;
status = BCryptOpenAlgorithmProvider(&hAlg,
GetPreferredHmacAlg(),
NULL,
BCRYPT_ALG_HANDLE_HMAC_FLAG)))
```

Volání `GetPreferredHmacAlg` není funkce CNG, jedná se o funkci, která slouží k získání preferovaného základního algoritmu HMAC, třeba z konfiguračního nastavení.

Generování náhodných čísel

Vzhledem k tomu, že kód na generování náhodných čísel je velmi malý, vložili jsme jej sem celý:

```
BCRYPT_ALG_HANDLE hRngAlg = NULL;
if (BCryptOpenAlgorithmProvider(&hRngAlg,
                                BCRYPT_RNG_ALGORITHM,
                                NULL,
                                0) == STATUS_SUCCESS) {
      BYTE buf[32];
      if (BCryptGenRandom(hRngAlg,
                          buf,
                          sizeof buf,
                          0) == STATUS_SUCCESS) {
          // Máme náhodná data.
      }

      BCryptCloseAlgorithmProvider(hRngAlg,0);
      hRngAlg = NULL;
}
```

CNG a FIPS

Federální standardy zpracování informací (Federal Information Processing Standards, FIPS) na adrese *http://www.itl.nist.gov/fipspubs/* definují standardy a návody, které vyvíjí Národní institut pro standardy a technologii (National Institute of Standards and Technology, NIST) pro federální počítačový systém USA. Pro tuto kapitolu lze využít pět následujících standardů:

- FIPS 140-2: Bezpečnostní požadavky pro kryptografické moduly.
- FIPS 180-2: Standard bezpečného hašování (Secure Hash Standard, SHS).
- FIPS 186-2: Standard digitálního podpisu (Digital Signature Standard, DSS).

- FIPS 197: Standard pokročilého šifrování (Advanced Encryption Standard, AES).
- FIPS 198: Autentizační kód zpráv hašovaný pomocí klíče (The Keyed-Hash Message Authentication Code, HMAC).

Tyto standardy definují kryptografické požadavky a kryptografické algoritmy, které se mají používat ve federálních informačních systémech USA. Následujícím postupem lze nastavit Windows Vista, aby používaly pouze algoritmy podléhající standardům FIPS:

1. Otevřete MMC.
2. Přidejte snap-in Group Policy Objects.
3. Přejděte postupně do položek Local Computer Policy, Computer Configurations, Windows Settings, Security Settings, Local Policies, Security Options.
4. Povolte následující volbu: „System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing." (Systémová kryptografie: Pro šifrování, hašování a podepisování používat algoritmy, jež jsou v souladu s FIPS.)

A již je tu zádrhel. Toto nastavení ovlivňuje pouze sestavy protokolů používané v SSL/TLS a v kódu .NET. Následující sekvence kódu v jazyce C# selže a vypíše výjimku `System.InvalidOperationException`, neboť používá algoritmus MD5, jenž není v souladu s FIPS.

```
MD5CryptoServiceProvider hash = new MD5CryptoServiceProvider();
byte[] result = hash.ComputeHash(ASCIIEncoding.UTF8.GetBytes(message));
```

V aplikaci CNG je možné určit, zdali je zapnuto vyžadování FIPS pomocí funkce `BCryptGetFipsAlgorithmMode`.

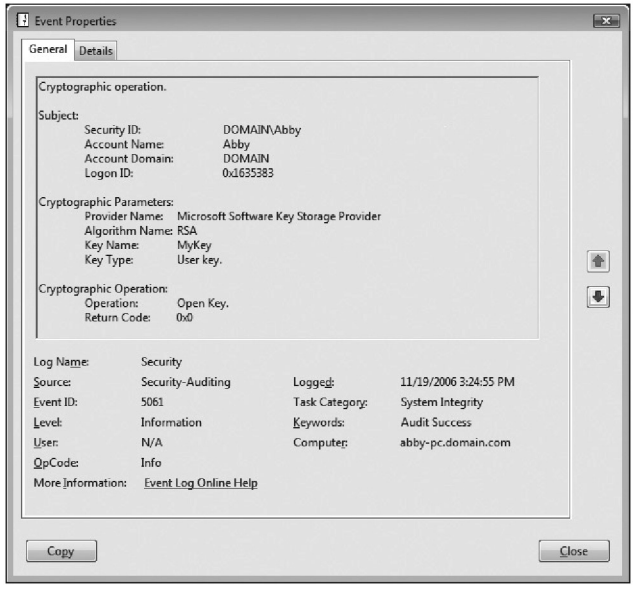
Vylepšené auditování

Ke splnění určitých požadavků v předpisech Common Criteria napomáhá audit rozmanitých operací s klíči. Následujícím příkazem vloženým do povýšeného příkazového řádku nastavíte audit klíčů ve Windows Vista.

```
auditpol /set /subcategory:"other system events" /success:enable /failu-
re:enable
```

Poznámka: Dokument The U.S. Government Protection Profile for Single-level Operating Systems in Environments Requiring Medium Robustness v1.67, §5.1 definuje požadavky bezpečnostního auditu, včetně těch, které se vztahují k používání kryptografických klíčů (NSA 2003).

Operace s klíči, například jejich vytvoření, mazání a přístup ke klíči, vyvolávají události, které ukazuje obrázek 7.1.



Obrázek 7.1 Auditovaná událost, k níž dojde, když CNG přistupuje ke klíči RSA

Záznamy si lze prohlédnout v bezpečnostním protokolu v prohlížeči událostí ve Windows (Windows Event Viewer).

Co v CNG chybí

Podobně jako CAPI, i CNG postrádá funkci vytvoření klíče pomocí hesla (RFC 2898). V zásadě byste nikdy neměli používat heslo přímo na šifrování dat; namísto toho je vhodné odvodit finální klíč z původního hesla. To obvykle znamená tisíce volání (počet iterací) kryptografické operace, například hašování, na původním klíči. O počtu iterací lze smýšlet jako o „kompenzátoru Mooreova zákona". Jak se počítače zrychlují, zvyšujeme zkrátka počet iterací, abychom vykompenzovali rychlost počítačů a zpomalili útočníka.

.NET Framework obsahuje podporu PBKDF, například třídu `Rfc2898DeriveBytes`. Zde je ukázka kódu, jenž tuto třídu používá:

```
string password = args[0];
byte[] salt = new byte[16];
new RNGCryptoServiceProvider().GetBytes(salt);
Rfc2898DeriveBytes pdb = new Rfc2898DeriveBytes(password, salt, 50000);
byte[] key = pdb.GetBytes(16);
byte[] iv = pdb.GetBytes(16);
Console.WriteLine("Klíč: " + Convert.ToBase64String(key));
Console.WriteLine("IV : " + Convert.ToBase64String(iv));
```

CNG obsahuje velmi flexibilní funkci výpočtu klíče, `BCryptDeriveKey`, kterou lze použít na vytvoření klíče stejným způsobem, jako to dělá SSL3, TLS1 a CMS, avšak není to

funkce, která převádí heslo na klíč (password-to-key). Doufejme, že se tento požadavek dočká splnění v budoucí verzi CNG.

Vylepšení SSL/TLS

V SSL/TLS došlo ve Windows Vista ke dvěma zásadním vylepšením. Prvním jsou nové šifrovací sestavy včetně podpory algoritmů Suite B: šifrování pomocí eliptických křivek, SHA-2 a AES. Windows Vista podporují šifrovací sestavy z tabulky 7.3.

Tabulka 7.3 Nové šifrovací sestavy SSL/TLS ve Windows Vista

TLS_RSA_WITH_AES_128_CBC_SHA	TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_RC4_128_SHA	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_RC4_128_MD5
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P256	SSL_CK_RC4_128_WITH_MD5
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P384	SSL_CK_DES_192_EDE3_CBC_WITH_MD5
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P521	TLS_RSA_WITH_NULL_MD5
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P256	TLS_RSA_WITH_NULL_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P384	TLS_RSA_WITH_DES_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P521	TLS_DHE_DSS_WITH_DES_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P256	TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P384	TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P521	TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P256	TLS_RSA_EXPORT_WITH_RC4_40_MD5
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P384	SSL_CK_DES_64_CBC_WITH_MD5
T LS_ECDHE_RSA_WITH_AES_256_CBC_SHA_P521	SSL_CK_RC4_128_EXPORT40_WITH_MD

SSL/TLS ve Windows Vista podporuje také následující šifrovací sestavy, které lze však použít pouze v kódu volajícím přímo poskytovatele protokolu SSL/TLS (například TLS1SP_NAME).

- TLS_RSA_WITH_NULL_MD5.
- TLS_RSA_WITH_NULL_SHA.

Jak jsme si již řekli dříve v této kapitole, lze také přidat vlastní šifrovací sestavu, pokud si napíšete modul (plug-in) poskytující SSL/TLS.

Druhé vylepšení spočívá v tom, že Windows Vista usnadňují konfiguraci šifrovacích sestav, které chcete v SSL/TLS podporovat; jde o prosté nastavení pravidla. Takové pravidlo nastavíte následovně:

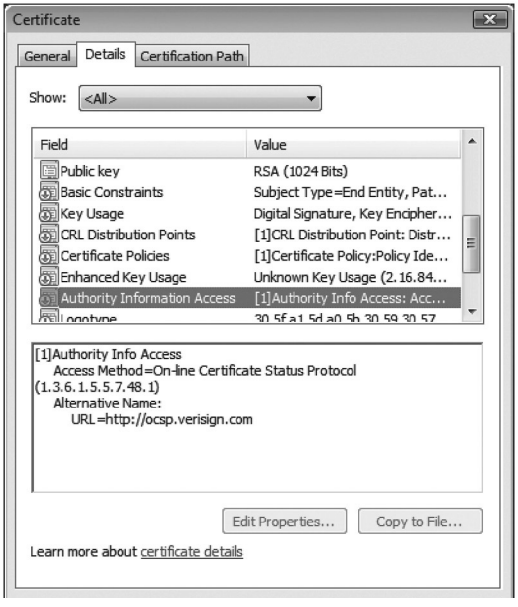
1. Otevřete mmc.exe.
2. Klepněte na položku File a poté na Add/Remove Snap-in.
3. Odrolujte dolů a zvolte Group Policy Object Editor.
4. Klepněte na tlačítko Add.
5. Klepnutím na tlačítko Browse zvolte vzdálený počítač nebo tlačítkem Finish lokální počítač.
6. Klepněte na tlačítko OK a poté ještě jednou na OK.
7. Otevřete následující uzly: Local Computer Policy, Computer Configuration, Administrative Templates a poté Network.
8. Klepněte na uzel SSL Configuration Settings.
9. Poklepejte na nastavení SSL Cipher Suite Order.
10. Klepnutím na tlačítko Explain zjistíte, jak lze toto nastavení upravit.

Ověření zrušení SSL/TLS a OCSP

Kontroly seznamu zrušených certifikátů (Certificate Revocation List, CRL) jsou dostupné ve všech dnes podporovaných verzích Windows. Když chce aplikace ověřit, zdali daný certifikát není zrušen, načte z certifikátu distribuční bod seznamu zrušených certifikátů (Certificate Revocation List Distribution Point, CDP). Vyhledávání v CRL může k načtení seznamu používat HTTP, LDAP nebo prostý přístup ke vzdálenému souboru. CRL je digitálně podepsaný seznam, jenž obsahuje sériová čísla zrušených certifikátů. Jestliže se sériové číslo ověřovaného certifikátu nalézá v CRL, je certifikát zrušen a aplikace by měla přijmout příslušná opatření. Windows Vista také podporují online protokol stavu certifikátu (Online Certificate Status Protocol, OCSP), definovaný pomocí RFC 2560 (RFC 2560). OCSP má oproti používání seznamů CRL své výhody i nevýhody, neboť tyto seznamy mohou být rozsáhlé, kdežto vyhledávání v OCSP je krátké. Hlavní nevýhodou práce s OCSP je, že aplikace musí být online, kdežto seznamy CRL lze cacheovat, což znamená, že vyhledávání v CRL nemusí nezbytně probíhat vzdáleně.

Poznámka: Windows Server 2008 podporuje server OCSP.

Skutečně dobrou zprávou je, že podpora OCSP je ve Windows Vista naprosto transparentní, pokud ověřovaný certifikát obsahuje příslušnou adresu URL pro přístup k informacím od autority (Authority Information Access, AIA). Podobně jako CDP, AIA může obsahovat adresu či adresy URL, pomocí nichž lze zjišťovat, zdali byl certifikát zrušen či nikoliv. Na obrázku 7.2 je znázorněna platná informace AIA na serverovém certifikátu.

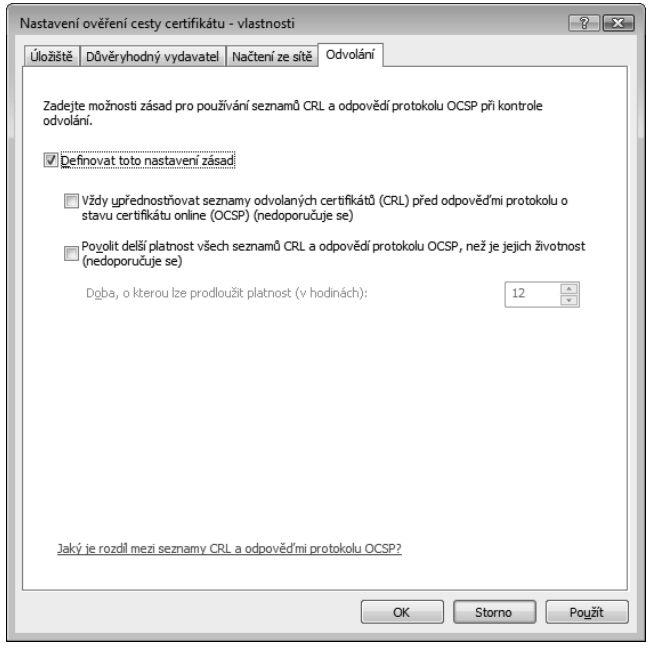


Obrázek 7.2 Certifikát Verisign, jenž obsahuje AIA pro použití v OCSP

Pokud vaše aplikace volá pro ověření certifikátu funkci `CertGetCertificateChain` a certifikát obsahuje příslušné AIA, Windows Vista poté použijí OSCP k ověření certifikátu ještě před prověřováním, zdali je v certifikátu distribuční bod CDP. Toto pravidlo lze změnit následujícím způsobem:

1. Otevřete mmc.exe.
2. Přidejte snap-in Group Policy Object Editor.
3. Otevřete uzel Local Computer Policy.
4. Otevřete Computer Configuration.
5. Otevřete Windows Settings.
6. Otevřete Security Settings.
7. Klepněte na uzel Public Key Policies.
8. Poklepejte na typ objektu Certificate Path Validation Settings Properties.
9. Klepněte na záložku Revocation.

Obrázek 7.3 ukazuje dialog revokačního pravidla.



Obrázek 7.3 Nastavení pravidla pro zrušený certifikát ve Windows Vista

Aplikaci se nemusí zdařit volání adresy URL pro CDP nebo OCSP, a v takovém případě funkce, jako je `CertGetCertificateChain`, selžou a vrátí chybu CRYPT_E_REVOCATION_OFFLINE. Je-li vaše aplikace vázaná, co by měla udělat, jestliže nemůže ověřit, je-li certifikát odvolaný či nikoliv? Neumíme vám dát stoprocentní odpověď, neboť ta závisí na povaze vaší aplikace. Jde-li o citlivou aplikaci nebo o aplikaci běžící v citlivém prostředí, měli byste operace související s používáním certifikátu zrušit.

Nástroj příkazového řádku `certutil -url` lze použít na interaktivní ověření certifikátu pomocí CRL neb OCSP.

Poznámka: Významné události týkající se certifikátu se archivují v prohlížeči událostí (Event Viewer). Otevřete prohlížeč událostí a dále protokoly aplikací a služeb (Application and Services Logs), následně otevřete uzel Microsoft, pak Windows a na závěr CAPI2.

Kořenové certifikáty ve Windows Vista

Během čisté instalace Windows Vista se nainstaluje jen velmi málo kořenových certifikátů. Ve skutečnosti se nainstalují pouze ty kořenové certifikáty, které jsou nutné ke startu operačního systému. Ve výchozím nastavení platí, že není-li kořenový certifikát důvěryhodný, Windows Vista pomocí Windows Update (WU) zjistí, patří-li daný kořenový certifikát mezi zhruba stovku certifikátů, které se instalovaly s Windows XP. Existuje-li tento certifikát na WU, operační systém jej zkopíruje a bezobslužně nainstaluje. Ale co když tento postup nechcete? Co když chcete věřit jen těm kořenovým certifikátům, kte-

rým opravdu důvěřujete? Vše, co je nutné provést, je zakázat funkci automatické aktualizace kořenových certifikátů ve Windows Vista, k čemuž slouží tento postup:

1. Otevřete mmc.exe a přidejte snap-in Local Computer Policy.
2. Přejděte na položku Computer Configuration → Windows Settings → Security Settings → Public Key Policies.
3. Klepněte na objekt Certificate Path Validation Settings.
4. Klepněte na záložku Network Retrieval.
5. Klepněte na nastavení Define these policy.
6. Zrušte volbu Automatically update certificates in the Microsoft Root Certificate Program.

Zrušené kryptografické funkce ve Windows Vista

Následující kryptografické funkce byly ve Windows Vista zrušeny nebo již nejsou podporovány:

- Objekty COM `XEnroll` a `SCdEnrl` již nejsou součástí Windows Vista; namísto nich byste měli používat rozhraní API `CertEnroll` (Microsoft 2006b).
- Počínaje Windows Vista již není podporováno rozhraní COM pro CryptoAPI s názvem `CAPICOM`. Potřebujete-li používat API na vyšší úrovni, měli byste použít .NET Framework.
- Na ověření certifikátů by se již neměla používat funkce `WinVerifyTrust`. Namísto ní používejte funkci `CertGetCertificateChain`.

Realizace

- Prověřte svou kryptografickou funkcionalitu a stanovte, zdali potřebujete algoritmy Suite B či nikoliv. Pokud ano, poslouží vám CNG.
- Určete, jak velké množství vašeho kódu používá přímé názvy algoritmů, začněte kód přepracovávat, aby načítal informace z konfiguračního úložiště, a poté zapište informaci o nastavení kryptografie do metadat dokumentu.
- Vyhodnoťte, které slabé algoritmy by bylo vhodné z vašeho kódu odstranit a nahradit je silnými či ještě silnějšími algoritmy a zároveň tak do vašeho kódu přidat kryptografickou pružnost.
- Některé kryptografické funkce již byly z Windows Vista odstraněny a měli byste přejít na používání novějších a robustnějších funkcí.

Literatura

(NSA 2005) National Security Agency. „Fact Sheet NSA Suite B Cryptography", *http://www.nsa.gov/ia/industry/crypto_suite_b.cfm*.

(Microsoft 2000) „Microsoft Kernel Mode Cryptographic Module", *http://www.microsoft.com/technet/archive/security/topics/issues/fipsdrsp.mspx*. 2000.

(NIST 2006) Computer Security Resource Center. „Cryptographic Toolkit: Secure Hashing", *http://csrc.nist.gov/CryptoToolkit/tkhash.html*.

(Anderson 1999) Serpent Homepage. *http://www.cl.cam.ac.uk/~rja14/serpent.html*.

(Microsoft 2006a) CNG Software Development Kit, *http://www.microsoft.com/downloads/details.aspx?FamilyID=&DisplayLang=en*.

(NSA 2003) National Security Agency. „U.S. Government Protection Profile Single Level Operating Systems for Medium Robustness Environments", *http://www.niapccevs.org/pp/draft_pps/pp_draft_slos_mr_v1.67.cfm*. 2003.

(RFC 2898) RSA Laboratories. Network Working Group, Request for Comments 2898. „PKCS #5: Password-Based Cryptography Specification Version 2.0", *http://www.ietf.org/rfc/rfc2898.txt*. 2000.

(RFC 2560) Entrust Technologies. Network Working Group, Request for Comments 2560. „X.509 Internet Public Key Infrastructure Online Certificate Status Protocol±OCSP", *http://www.ietf.org/rfc/rfc2560.txt*. 1999.

(Microsoft 2006b) Certificate Enrollment API, *http://msdn2.microsoft.com/en-gb/library/aa374863.aspx*.