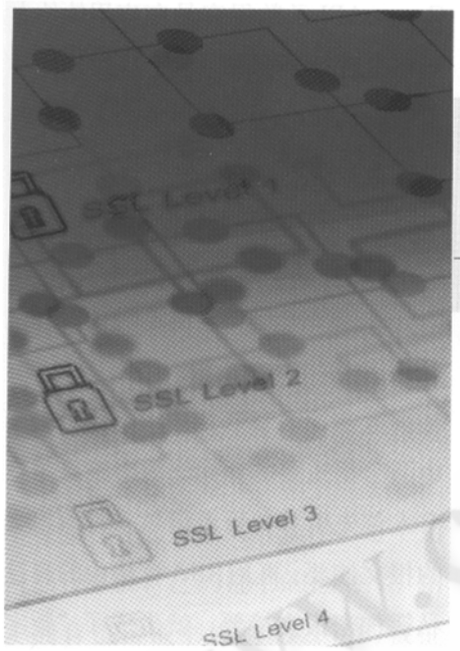
基于Microsoft密码体系的信息安全的实现



马 军 周艳梅 (西安卫星测控中心软件室 710073)

摘要 本文描述了Microsoft加密技术、证书以及证书的管理和使用，用CryptoAPI开发信息安全的程序方法和步骤，最后给出VC++开发加密和数字签名程序的实例。

关键词 CryptoAPI 加密 证书

计算机网络从问世到今天，其技术的发展之快，是人们始料不及的。人们对网络的依赖性越来越强，网上传输敏感信息越来越多,如信用卡、银行帐号和订购信息等。不仅要求网络能连续的工作，而且还要保证网上信息传输的安全性，即：保密性、认证、完整性、不可否认性、不可伪造性。由于Internet及其TCP/IP协议等最初设计是非商业用途的，几乎没有考虑网络安全性，如何实现信息的安全以成为人们关心的热点之一。

1 Microsoft的密码体系和CryptoAPI的编程基础

利用Microsoft的密码系统开发信息安全程序主要涉及密码服务提供者（cryptographic service provider，CSP）和密码应用程序接口CryptoAPI（Cryptographic Application Programming Interface）。在Microsoft的密码体系中加密算法封装在CSP内，密码服务提供者(CSP)负责创建和删除密钥，通过密钥，执行事实上的加密操作。它可使用户在对网络加密机制和加密算法不太了解的情况下完成安全程序的开发。为Win32应用程序提供了认证、编码、加密和签名等信息安全处理。CryptoAPI编程模型如图1。

该模型同系统的图形设备接口类似，其中密码服务提供者等同于图形设备驱动程序，上层的应用程序，都不同硬件直接打交道，该模型由应用程序、CryptoAPI、CSP以及操作系统和硬件层等部分组成。CryptoAPI信息的加密和解密由简单消息函数、底层消息函数和基本加密函数组成，证书的使用和管理由证书的加密/解密函数、证书库的管理函数组成；简单消息函数位于底层消息函数的上层，是由多个底层消息函数和证书函数封装成一个简单消息函数，使用更方便。底层消息函数完成信息的加密和解密，也可完成信息签名和验证。基本加密函数直接同密码服务提供者进行通信，所有加密解密处理都有CSP完成，系统可以有一个或多个密码服务提供者，不同的密码服务提供者提供的加密解密支持。证书的加密解密函数，完成证书的加密和解密功能，证书库的管理函数完成证书的获取、存储、更新、列举和验证。

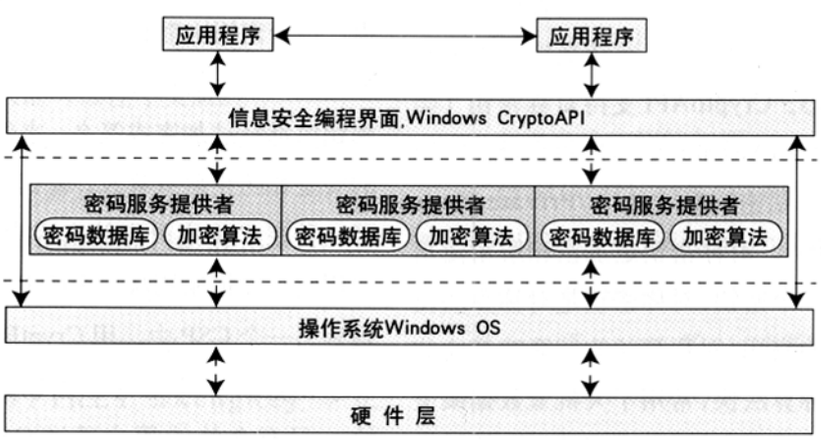


图 1 CryptoAPI编程模型

一个CSP最少包括一个dll库和一个签名文件，签名文件定期验证签名，可使CryptoAPI识别CSP并验证其完整性，CSP通过Cpredll.dll与应用程序通信，不同的CSP提供不同的加密支持，提供不同的加密算法如：DES、RC2、RC4等。每个CSP都有一个与之相关联的密钥数据库（key dabase），该数据库包含一个或多个密钥容器（key container），每个密钥容器包含属于特定用户私钥和公钥。每个密钥容器都有一个唯一的名字，若没有该数据库，CryptoAPI无法工作，用户、密钥数据库、CSP、密钥容器之间的关系如图2。

2 Microsoft 的加密技术

信息安全可以通过利用对称加密钥和非对称密钥加密和解密来实现，CryptoAPI通过连接CSP，密钥和密钥管理（产生密钥，交换密钥），加密和解密以及数字签名和验证，实现信息的安全。

利用CryptoAPI开发加密软件，首先要同CSP建立连接。在典型的CryptoAPI应用中，应用软件第一个调用的函数是CryptAcquireContext，该函数返回特定的CSP句柄，该句柄指定特定CSP中特定的密钥容器。该句柄用于特定CSP上做后续的函数的调用。

密钥（Keys）和密钥管理是保证信息安全的关键，密钥是信息加密的核心。CryptoAPI支持对称密钥（会话密钥 Session keys）和非对称密钥（公/私密钥对 Public/Private Keys Pairs），对称密钥是加密和解密使用同一个密钥，对称密钥是对称算法使用的密钥，对称算法的计算速度比非对称算法快，常用于大批量数据或文件的加密和解密。常用的对称算法有DES、RC4等。它的关键是密钥的管理。会话密钥存在于CSP内部，且经常改变，来保证会话密钥的机密性。用CryptDeviceKey和CryptGenKey函数创建密钥，其中CryptDevice函数是根据口令形成密钥，CryptGenKey产生随机密钥。非对称密钥（公开/私有密钥）是加密和解密使用不同的密钥。在每个用户的密钥容器中都有交换密钥对（exchange key pair）和签名密钥对(signature key pair)，其中交换密钥是用于加密会话密钥。签名密钥是利用哈希技术产生报文摘要形成签名时使用。每个CSP都有一个存放密钥的密钥库，每个密钥库包含一个或多个密钥容器，每个容器包含属于特定用户的密钥对。它们之间的关系如图2。

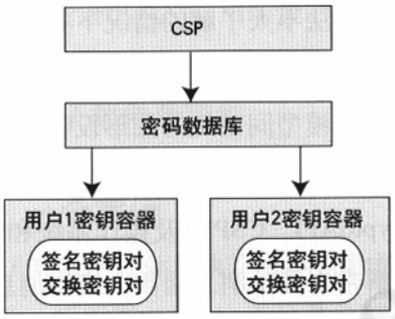


图2 用户、密钥库、CSP、密钥容器之间的关系图

若用户要存储会话密钥以便以后使用，或给其他用户发送密钥，就要将密钥从CSP中导出，导出的密钥都存放在二进制形式的密钥块(key blob)中。每个密钥块都有一个标准头，下来是表示密钥的不可读的信息，每个密钥块都经过加密或签名，来保证它的安全和防止恶意篡改。密钥块提供了一种在CSP之外存储密钥的方法，利用它可以把密钥从一个CSP安全传输到另一个CSP中。用CryptExportKey函数从CSP中导出密钥到密钥块。只有合法的用户通过CryptImprotKey函数使用该密钥块。密钥块主要分为三类：简单密钥块(Simple key blobs)、公钥密钥块(Public key blobs)和私钥密钥块(Private key blobs)。

加密/解密的算法封装在CSP中，使用对称算法来加密信息，用CryptoAPI函数很容易在应用软件中对信息进行加密和解密。应用软件用函数CryptEncrypt用指定的密钥对明文加密，通过CryptDecrypt函数用指定的密钥对密文解密，加密算法在创建密钥时指定，也可通过调用CryptSetKeyParam来指定附加的加密参数。

数字签名和哈希技术是信息安全中两个很重要的概念，数字签名是签字人首先用单向哈希函数求得原报文的报文摘要，再用RSA算法中的自己私钥加密报文摘要，就生成了“数字签名”，接收方用签字人的公钥解密“数字签名”，得到签字人发来的报文摘要，再计算收到报文的报文摘要，将两个报文摘要比较，若相同，则验证了报文及签名是真实完整的。哈希技术是通过哈希函数的单向不可逆性实现的，单向哈希函数算法是将任意长度的输入报文，经计算，得出固定位的输出，称为报文摘要。所谓单向是指该算法是不可逆的，故只要对原报文稍做修改，就会得出截然不同的报文摘要。由于哈希算法的单向性和严密性，接收方可确保没有其他人能够生成与收到的报文摘要相同的新消息或原始报文的签名，由于RSA算法的坚固性，接收方可确保只有私钥的拥有者才能生成签名，从而验证了发送方的身份，实现不可否认性，从而实现了数字签名。不同CSP支持不同的哈希算法，如MD2、MD5和SHA1，CryptoAPI提供相应的函数完成哈希、签名和验证。

3 Microsoft 证书的管理和使用技术

证书是网络中身份认证的主要途径。在信息安全中使用证书涉及认证中心(Certificate Authority, CA)、证书服务器(Certificate Server)、证书的请求(Certificate Request)、证书的列表(Certificate List)，以及证书的标准等。证书由CA来颁发，并带有用户的公钥和CA私钥做的数字签名等，目前证书一般使用的是X.509标准。

微软的证书服务器（Microsoft Certificate Server，MCS）的体系结构如图3，中间模块接收证书请求，并把请求提交给作为MCS的核心服务器引擎，所有证书都在证书库中，证书一般都存储到永久介质中，每个用户都有存放自己的证书的证书库，用户拥有包含证书机构的公钥的根证书，根证书一般存放在注册表中ROOT库中。证书库中的证书以单向链表的形式存储，即每个证书库有一个指向第一个证书的指针，每个证书有一个指向下一个证书的指针，最后一个指向NULL。

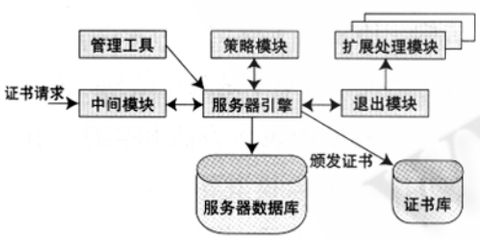


图 3 微软证书服务器（MCS）的体系结构

CryptoAPI提供了证书使用和管理函数，包括证书加密/解密函数(certificate encode/decode functions)、证书库函数（Certificate Store Functions）和证书辅助函数(Certificate Helper Functions)，证书库函数有包括公共函数（用于证书的存储和更新），可以管理逻辑或虚拟库（Logical store or virtual store）、远程库(Remote store)、外部库(External store)和重定位库(Relocated store)等。证书辅助函数主要包括证书的比较、转换、签名、哈希和验证。

管理证书一般经历打开证书库、查找证书、增加、删除和修改、保存证书最后关闭证书库等步骤。使用证书一般经历打开证书库，证书以及证书属性的列表，利用证书进行身份认证、加密、解密数字签名和验证等安全处理，最后关闭证书库步骤。

4 一个实例

发方对明文加密，同时对明文的摘要进行签名，而后把密钥、密文、和签名发给接收方，接收方对收到的密文解密，最后验证签名。采用RC2算法加密信息，用MD5哈希算法产生报文摘要，进行数字签名，来保证信息的机密性、可人证性、完整性和不可否认性。用CryptExportKey导出密钥，CryptImportKey导入密钥，CryptEncrypt对明文加密同时产生报文摘要，CryptDencrypt对密文解密，CryptSignHash数字签名，CryptVerifySignature验证签名。明文加密和数字签名程序如下，为节省篇幅，在程序中认为加密密钥是随机产生的，对pbBuffe的内容进行加密和签名，没有进行错误处理。解密与签名的认证与该程序类似。

```
void CCryptsignDlg::Onencryptsign()
{ //定义并初始化
    HCRYPTPROV hProv   = 0,
HCRYPTHASH hHash  = 0;
    HCRYPTKEY hKey     = 0,
HCRYPTKEY hXchgKey = 0,
HCRYPTKEY hKeysign = 0;
    //定义并初始化密钥块
    PBYTE pbKeyBlob = NULL,
PBYTE pbKeyBlobsign = NULL;
    DWORD dwKeyBlobLen;
    PBYTE pbSignature,  DWORD
dwBlobLensign;// 签名密钥长度
    DWORD dwCount,DWORD
dwSigLen, LPTSTR  szDescription
=NULL;
    BYTE *pbBuffer = (BYTE *)"The
data that is to be signed and encrypted";
    DWORD dwBufferLen =strlen
((char *)pbBuffer)+1;
    // 获取缺省的CSP的句柄.
    CryptAcquireContext(&hProv,
NULL, NULL, PROV_RSA_FULL, 0);
    // 创建哈希对象。
    CryptCreateHash(hProv,
CALG_MD5, 0, 0, &hHash);
    // 获取签名密钥句柄
    CryptGetUserKey(hProv,
AT_SIGNATURE,&hKeysign);
    CryptExportKey(hKeysign,NULL,
PUBLICKEYBLOB,0,NULL,
&dwBlobLensign);
    // 确定BLOB大小并分配缓冲区
    pbKeyBlobsign = (BYTE *) malloc
(dwBlobLensign);
    CryptExportKey(hKeysign,NULL,
PUBLICKEYBLOB,0,pbKeyBlobsign,
&dwBlobLensign);
    // 创建随机会话密钥.
    CryptGenKey(hProv, CALG_RC2,
CRYPT_EXPORTABLE, &hKey);
    // 获得交换密钥的句柄
    CryptGetUserKey(hProv,
AT_KEYEXCHANGE, &hXchgKey);
    // 确定BLOB大小并分配缓冲区
    CryptExportKey(hKey,
hXchgKey, SIMPLEBLOB, 0, NULL,
&dwKeyBlobLen);
```

（下转第30页）

(上接第36页)

```
pbKeyBlob = (BYTE *)malloc(dwKeyBlobLen);
// 导出会话密钥
CryptExportKey(hKey, hXchgKey, SIMPLEBLOB, 0, pbKeyBlob, &dwKeyBlobLen);
// 分配缓冲区
pbBuffer =(BYTE *) malloc(dwBufferLen);
// 加密信息同时产生报文摘要
dwCount = dwBufferLen;
CryptEncrypt(hKey, hHash, 0, 0, pbBuffer, &dwCount, dwBufferLen);
CryptSignHash(hHash, AT_SIGNATURE, szDescription, 0, NULL, &dwSigLen);
// 分配缓冲区.
pbSignature = (BYTE *) malloc(dwSigLen);
// 签名
CryptSignHash(hHash, AT_SIGNATURE, szDescription, 0, pbSignature, &dwSigLen);
MessageBox("OK");
// 释放密码服务提供者句柄.
CryptReleaseContext(hProv, 0);
// 释放内存，删除密钥和句柄
if(pbKeyBlob) free(pbKeyBlob);
if(pbBuffer) free(pbBuffer);
if(hKey) CryptDestroyKey(hKey);
if(hXchgKey) CryptDestroyKey(hXchgKey);
if (hKeysign) CryptDestroyKey(hKeysign);
if(hHash) CryptDestroyHash(hHash);
if(hProv) CryptReleaseContext(hProv, 0);}
```

5 结语

本文描述了在Microsoft密码体系下用CryptoAPI开发信息安全的程序方法和步骤，利用CryptoAPI可使用户在对网络加密机制和加密算法不太了解的情况下完成信息安全程序的开发，为Win32应用程序提供了认证、编码、加密和签名等信息安全服务。■

参考文献

1 *MSDN Library Visual Studio 6.0。*

2 *www.microsoft.com/security*