功能：连接CSP，获得指定CSP的密钥容器的句柄；

原型：

BOOL WINAPI CryptAcquireContext(

  \_\_out  HCRYPTPROV \*phProv,    //CSP句柄指针

  \_\_in   LPCTSTR pszContainer,       //密钥容器名称，指向密钥容器的字符串指针；

         //如果dwFlags为CRYPT\_VERIFYCONTEXT，pszContainer必须为NULL

  \_\_in   LPCTSTR pszProvider,          //指向CSP名称的字符串指针；

//为NULL，表示使用默认的CSP

  \_\_in   DWORD dwProvType,       //CSP类型

  \_\_in   DWORD dwFlags         //标志位

);

其中，dwFlags取值如下所示：

CRYPT\_VERIFYCONTEXT---指出应用程序不需要使用公私钥对，如程序只执行哈希和对称加密，只有程序需要创建签名和解密消息时才需要访问私钥

CRYPT\_NEWKEYSET---使用指定的密钥容器名称创建一个新的密钥容器；如果pszContainer为NULL，密钥容器就使用缺省的名称创建

CRYPT\_MACHINE\_KEYSET---由此标志创建的密钥容器只能由创建者本人或系统管理员身份的人使用

CRYPT\_DELETEKETSET---删除由pszContainer指定的密钥容器；如果pszContainer为NULL，缺省名称的容器就会被删除。此容器里的所有密钥对也会被删除

CRYPT\_SILENT---应用程序要求CSP不显示任何用户界面

看到这里，大家也许对CSP还比较迷惑。其实CSP是真正实行加密的独立模块，他既可以由软件实现也可以由硬件实现。但是他必须符合CryptoAPI接口的规范。

每个CSP都有一个名字和一个类型。每个CSP的名字是唯一的，这样便于CryptoAPI找到对应的CSP。目前已经有9种CSP类型，并且还在增长。下表列出出它们支持的密钥交换[**算法**](http://lib.csdn.net/base/31)、签名算法、对称加密算法和Hash算法。

(表一)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CSP类型 | 交换算法 | 签名算法 | 对称加密算法 | Hash算法 |
| PROV\_RSA\_FULL | RSA | RSA | RC2  RC4 | MD5  SHA |
| PROV\_RSA\_SIG | none | RSA | none | MD5  SHA |
| PROV\_RSA\_SCHANNEL | RSA | RSA | RC4  DES  Triple DES | MD5  SHA |
| PROV\_DSS | DSS | none | DSS | MD5  SHA |
| PROV\_DSS\_DH | DH | DSS | CYLINK\_MEK | MD5  SHA |
| PROV\_DH\_SCHANNEL | DH | DSS | DES  Triple DES | MD5  SHA |
| PROV\_FORTEZZA | KEA | DSS | Skipjack | SHA |
| PROV\_MS\_EXCHANGE | RSA | RSA | CAST | MD5 |
| PROV\_SSL | RSA | RSA | Varies | Varies |

从图一可以看到，每个CSP有一个密钥库，密钥库用于存储密钥。而每个密钥库包括一个或多个密钥容器（Key Containers）。每个密钥容器中含属于一个特定用户的所有密钥对。每个密钥容器被赋予一个唯一的名字。在销毁密钥容器前CSP将永久保存每一个密钥容器，包括保存每个密钥容器中的公/私钥对（见图二）。

返回值：操作成功返回TRUE，否则返回FALSE。调用GetLastError()可获得更多信息。