是一个独立的密码模块，不仅可以存储私钥，而且可以执行所有的密码操作。CSP 负责创建和销毁密钥并提供各种密码操作。不同 CSP 通过名称来区别，Windows 系统已安装的 CSP 包括：Microsoft Base Cryptographic Provider v1.0、Microsoft Enhanced Cryptographic Provider v1.0、Microsoft Strong Cryptographic Provider 等，这些 CSP 均由软件实现，属于软件密码模块。

Windows CryptoAPI 包含 CSP 访问函数，允许应用系统直接访问，主要包括以下 API 函数。

① CryptAcquireContext：根据 CSP 名称获得 CSP 句柄；
② CryptEnumProviders：枚举系统中所有 CSP；
③ CryptGetDefaultProvider：获得系统缺省 CSP；
④ CryptSetProvider：设置系统缺省 CSP；
⑤ CryptGetProvPara：获得 CSP 各种属性；
⑥ CryptSetProvParam：设置 CSP 各种属性；
⑦ CryptReleaseContext：释放 CSP 句柄。

大部分 Web 服务器已经支持数字证书机制，如 IIS、Apache、Tomcat 等，只有 Web 服务器申请并配置了 SSL 服务器证书，才能通过 TLS/SSL 服务，实现用户对网站的身份认证、用户与网站间数据的保密性、网站对用户的身份认证等安全功能。但大部分 Web 服务器自身私钥的存储方式和保存格式并不公开，完全由 Web 服务器进行安全管理，且只能通过专用工具或 UI 界面进行管理和配置，不允许其他应用系统进行访问。

GUOXU.NokiaMS

第 12 章　私钥与证书访问方式

12.1　CryptoAPI

12.1.1　CryptoAPI 简介

Windows CryptoAPI 是微软公司提出的安全加密应用服务框架，它提供了在 Win32 环境下使用认证、编码、加密和签名等安全服务的标准加密接口，用于增强应用程序的安全性与可控性。应用开发者在不了解复杂的加密机制和加密算法的情况下，可以简便、快速地开发出标准、通用和易于扩展的安全加密应用程序。

CryptoAPI 是一组函数，为了完成数学计算，必须具有密码服务提供者模块（Cryptographic Service Provider，CSP）。Microsoft 通过 RSA Base Provider 在操作系统级提供一个 CSP，使用 RSA 公钥加密算法，更多的 CSP 可以根据需要增加到应用中。事实上，CSP 有可能与硬件设备（如智能卡）一起来进行数据加密。CryptoAPI 接口允许通过简单的函数调用来加密数据、交换公钥、哈希一个消息来建立摘要以及生成数字签名。CryptoAPI 还为许多高级安全性服务提供了密码操作，包括用于加密客户机/服务器消息，用于在各个平台之间传递机密数据和密钥的 PFX、代码签名等。

CryptoAPI 体系共由 5 部分组成，体系结构如图 12-1 所示。

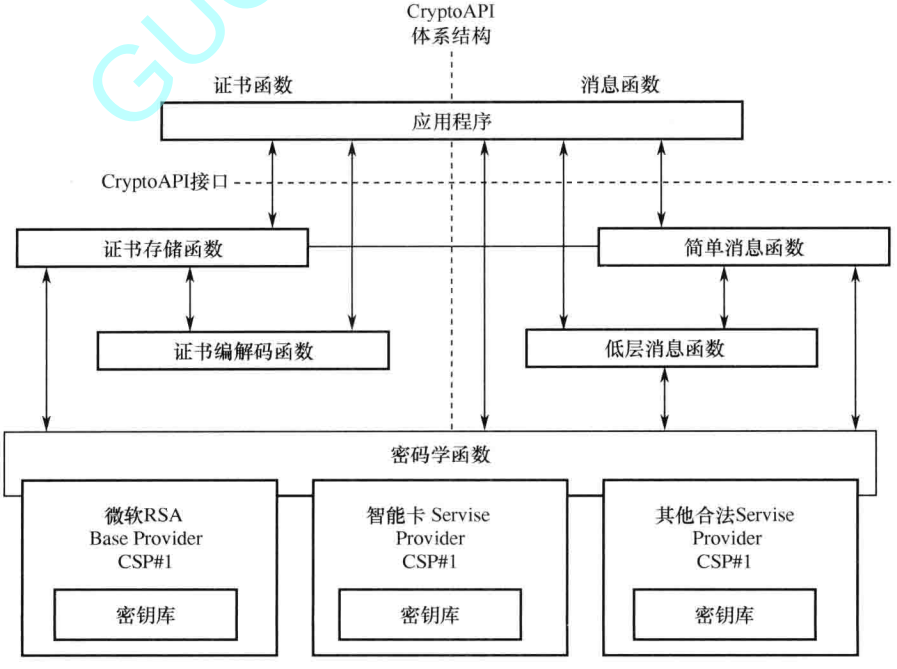

图 12-1　CryptoAPI 体系结构

GUOXUEKAIS

（1）基本加密函数（Base Cryptographic Functions）

用于选择 CSP、建立 CSP 连接、产生密钥、交换及传输密钥等操作。

（2）证书编解码函数（Certificate Encode/Decode Functions）

用于数据加密、解密、哈希等操作。这类函数支持数据的加密/解密操作；计算哈希、签发和验证数字签名操作；实现证书、证书撤销列表、证书请求和证书扩展等编码和解码操作。

（3）证书库管理函数（Certificate Store Functions）

用于数字证书及证书库管理等操作。这组函数用于管理证书、证书撤销列表和证书信任列表的使用、存储、获取等。

（4）简单消息函数（Simplified Message Functions）

用于消息处理，比如消息编码/解码、消息加/解密、数字签名及签名验证等操作。它把多个低层消息函数包装在一起以完成某个特定任务，方便用户的使用。

（5）低层消息函数（Low-level Message Functions）

低层消息函数对传输的 PKCS#7 数据进行编码，对接收到的 PKCS#7 数据进行解码，并且对接收到的消息进行解码和验证。它可以实现简单消息函数的所有功能，且提供更大的灵活性，但需要更多的函数调用。

每类函数的命名前缀都有约定，前缀约定如下：基本加密函数 Crypt、证书编码与解码函数 Cert、证书库管理函数 Store、简单消息函数 Message、低层消息函数 Msg。

CryptoAPI 之上是应用程序，之下是 CSP。CSP 是一个真正执行加密功能的独立模块，典型的 CSP 有微软 RSA Base Provider。任何一个加密服务提供者若想成为合法的 CSP，就必须获得微软授权的一个签名文件，该签名文件保证了该 CSP 的合法性和 CryptoAPI 能够识别它。CryptoAPI 使用系统注册表存储一个 CSP 数据库，CSP 数据库中记录了所有已安装到计算机中的 CSP。

每个 CSP 都有一个名字和一个类型，CSP 的名字是唯一的，这样便于 CryptoAPI 找到对应的 CSP。目前已经有 10 种 CSP 类型，并且还会继续增长。表 12-1 列出它们支持的密钥交换算法、签名算法、对称加密算法和哈希算法。

表 12-1　CSP 类型列表

CSP 类型	交 换 算 法	签 名 算 法	对称加密算法	哈 希 算 法
PROV_RSA_FULL	RSA	RSA	RC2、RC4	MD5、SHA
PROV_RSA_AES	RSA	RSA	RC2、RC4、AES	MD5、SHA
PROV_RSA_SIG	无	RSA	无	MD5、SHA
PROV_RSA_SCHANNEL	RSA	RSA	RC4、DES、3DES	MD5、SHA
PROV_DSS	DSS	无	DSS	MD5、SHA
PROV_DSS_DH	DH	DSS	CYLINK_MEK	MD5、SHA
PROV_DH_SCHANNEL	DH	DSS	DES、3DES	MD5、SHA
PROV_FORTEZZA	KEA	DSS	Skipjack	SHA
PROV_MS_EXCHANGE	RSA	RSA	CAST	MD5
PROV_SSL	RSA	RSA	可变	可变

每个 CSP 都有一个密钥库，里面存储着由 CSP 保存的算法密钥。每个密钥库都包含

一个或多个密钥容器，每个容器都包含一到多个密钥对。每个密钥容器都被赋予一个唯一的名字，这个名字是程序要获得此容器句柄时传给函数的参数。

12.1.2 使用证书

12.1.2.1 函数说明

1. 打开证书库

```
HCERTSTORE WINAPI CertOpenSystemStore(
    HCRYPTPROV hProv,
    LPCTSTR szSubsystemProtocol);
```

参数：

hProv：CSP 句柄，如果为 NULL，则为缺省 CSP；如果不为 NULL，必须是由 CryptAcquireContext 得到的 CSP 句柄。

szSubsystemProtocol：系统证书库的名称。可以为“CA”、“MY”、“ROOT”、“SPC”。

2. 枚举证书库证书

```
PCCERT_CONTEXT WINAPI CertEnumCertificatesInStore(
    HCERTSTORE hCertStore,
    PCCERT_CONTEXT pPrevCertContext);
```

参数：

hCertStore：证书存储区的句柄。

pPrevCertContext：上一个证书内容的 CERT_CONTEXT 结构指针，当第一次开始列举时参数必须为 NULL。

说明：从证书存储区中获取第一个或者下一个证书。使用循环，它可以获取存储区所有的证书序列。

3. 获取证书名称串

```
DWORD WINAPI CertGetNameString(
    PCCERT_CONTEXT pCertContext,
    DWORD dwType,
    DWORD dwFlags,
    void* pvTypePara,
    LPTSTR pszNameString,
    DWORD cchNameString);
```

参数：

pCertContext：包含 CERT_CONTEXT 证书内容的指针。

dwType：指定如何找到名字和输出什么格式。例如 CERT_NAME_SIMPLE_DISPLAY_TYPE 会输出找到的通用名、组织单位、组织结构、E-mail。

dwFlags：指定需要处理的类型，为 CERT_NAME_ISSUER_FLAG 时，请求颁发者名字，不设置时，请求使用者名字。

pvTypePara：由参数 dwType 类型指定，是一个指向 DWORD 或 OID 对象的指针。

GUOXU.nokaMS

pszNameString：存储转换后字符串序列。

cchNameString：pszNameString 分配的缓冲区长度，包括结尾的 NULL 字符。

说明：得到证书的使用者或颁发者名称，并且把它转换成字符串。

4．关闭证书库

```
HCERTSTORE WINAPI CertCloseStore(
    HCERTSTORE hCertStore,
    DWORD dwFlags);
```

参数：

hCertStore：已经打开的证书库句柄。

dwFlags：此参数的典型值为 0。缺省就是关闭证书库，对于为上下文分配的内存并不释放。如果想要检查并且释放所有为证书、CRL 和 CTL 上下文分配的内存，应设置下列标志：CERT_CLOSE_STORE_CHECK_FLAG（检查没有释放的证书、CRL 和 CTL 上下文）、CERT_CLOSE_STORE_FORCE_FLAG（强制释放所有和证书库相关的上下文）。

说明：此函数释放证书库句柄。

5．复制证书句柄

```
PCCERT_CONTEXT WINAPI CertDuplicateCertificateContext(
    PCCERT_CONTEXT pCertContext);
```

参数：

pCertContext：包含 CERT_CONTEXT 证书内容的指针。

说明：复制一份现有证书句柄。

6．释放证书句柄

```
BOOL WINAPI CertFreeCertificateContext (
    PCCERT_CONTEXT pCertContext);
```

参数：

pCertContext：包含 CERT_CONTEXT 证书内容的指针。

说明：释放证书句柄。

12.1.2.2　示例程序

证书存储在 Windows 证书库里，要对证书进行访问，先使用 CertOpenSystemStore 函数打开证书库，再使用 CertEnumCertificatesInStore 函数枚举证书，再使用 CertGetNameString 函数获取证书名称，使用完证书库后，使用 CertCloseStore 函数关闭，以防止内存泄露。

在查找到证书后，如果需要保存留作后用，需要使用 CertDuplicateCertificateContext 函数复制证书句柄。当不再使用此证书句柄时，使用 CertFreeCertificateContext 释放。

1．证书库操作

```
HCERTSTORE          hCertStore;
PCCERT_CONTEXT      pCertContext=NULL, tmpContext;
char                pszNameString[256];
```

GUO/U.nokiawe