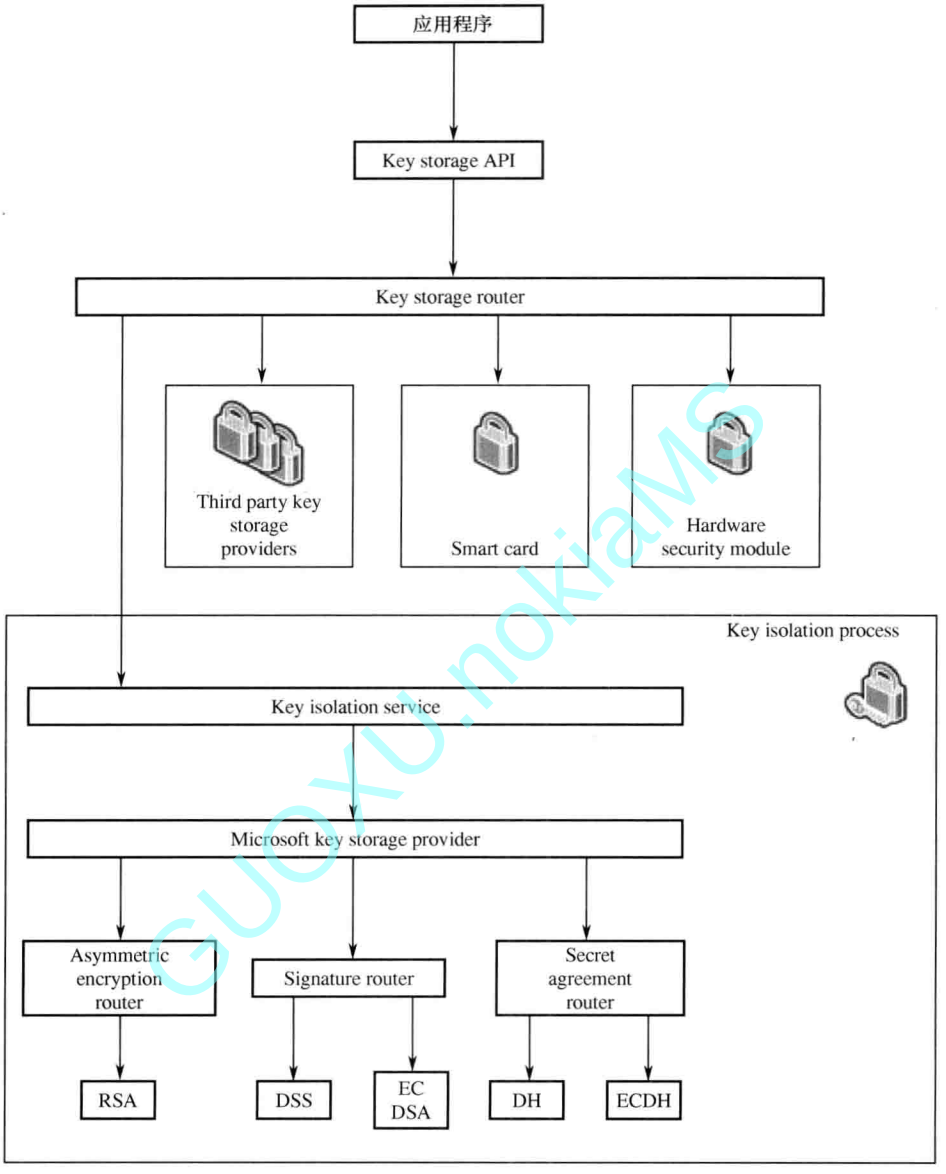

图 12-8　CNG 密钥隔离架构和功能

为最大限度地提高性能，使用 LSA 进程作为密钥隔离进程。私钥的访问都通过密钥存储路由，密钥存储路由输出了全面管理和使用私有密钥的函数。

CNG 存储私钥和公钥部分，密钥对的公共部分也由密钥隔离服务维护，并通过使用本地远程过程调用（LRPC）进行访问。密钥存储路由使用 LRPC 与密钥隔离进程交互。所有的私钥访问都要经过私钥路由，并由 CNG 审计。

CNG 支持以下类型的密钥：DH（Diffie-Hellman）、DSA（Digital Signature Algorithm）、RSA（PKCS#1）、部分 CryptoAPI 公私钥、ECC（Elliptic Curve Cryptography）。

与 CryptoAPI 相比，CNG 容器的不同之处如下。

① CNG 使用与 CSP（由 Rsaenh.dll 和 Dssenh.dll 创建）不同的密钥文件名，CSP 文件使用.key 扩展名，CNG 密钥文件无此扩展名。

② CNG 完全支持 Unicode 密钥容器名，CNG 使用 Unicode 密钥容器名的哈希值，而 CryptoAPI 使用 ANSI 密钥容器名的哈希值。

③ CNG 对 RSA 密钥对的支持更灵活，如 CNG 支持超过 32 位长度的公钥 E 参数，并支持不同长度的 P 和 Q 值。

④ CryptoAPI 密钥容器文件保存在与用户 SID 相同名字的目录中，CNG 中不再如此，这样可以防止用户从一个域转移到另一个域时丢失密钥。

⑤ CNG 的 KSP 和密钥名字的长度限制为 MAX_PATH 个 Unicode 字符，CryptoAPI 的 CSP 和密钥名字的长度限制为 MAX_PATH 个 ANSI 字符。

⑥ CNG 支持用户定义的密钥属性，用户可以创建和关联密钥的用户属性，这些属性可以和持久密钥存放在一起。

当持久化密钥时，CNG 创建 2 个文件。第一个文件（总是创建）包含 CNG 新格式密钥，此文件不能被 CryptoAPI 的 CSP 使用；第二个文件以 CryptoAPI 密钥容器方式保存同样的密钥，第二个文件符合 Rsaenh.dll 使用规范。第二个文件只有在调用 NCryptFinalizeKey 时设置 NCRYPT_WRITE_KEY_TO_LEGACY_STORE_FLAG 参数，且密钥类型是 RSA 时才创建，当密钥为 DH 或 DSA 时，不创建第二个文件。

当应用程序访问存在的持久密钥时，CNG 首先打开 CNG 格式文件。如果此文件不存在，它会尝试打开 CryptoAPI 密钥容器中的匹配密钥。

4. 算法提供程序

如何使用 CNG 来执行各种常见的加密操作呢？首先需要一个算法提供程序。BCrypt 定义的所有 CNG 对象均由 BCRYPT_HANDLE 标识，算法提供程序也不例外。BCryptOpenAlgorithmProvider 函数基于选择的算法和实现（可选）加载算法提供程序并返回一个句柄，以便在后续 CNG 函数调用中使用。不管是在用户模式下还是在内核模式下进行编码，BCrypt 还采用 Windows Driver Kit（WDK）中的 NTSTATUS 类型指示错误信息。下面介绍如何将算法提供程序加载到内存中：

```
BCRYPT_HANDLE algorithmProvider = 0;
NTSTATUS status = ::BCryptOpenAlgorithmProvider(
      &algorithmProvider, algorithmName,   implementation, flags);
if (NT_SUCCESS(status)) {
    // Use algorithm provider
}
```

在大多数情况下，将为 implementation 和 flags 参数传递 0。为 implementation 传递 0 表示应该为 algorithmName 参数标识的特定算法加载默认算法提供程序。NT_SUCCESS 宏用于指示状态值表示成功还是失败。

处理完算法提供程序后，必须通过将 BCryptOpenAlgorithmProvider 返回的句柄传递给 BCryptCloseAlgorithmProvider 函数将其卸载，如下所示：

```
status = ::BCryptCloseAlgorithmProvider( algorithmProvider, flags);
```

```
ASSERT(NT_SUCCESS(status));
```

当前没有为此函数定义 flags，因此必须为 flags 参数传递 0。

12.4.2　使用证书

在 CryptoAPI 中，可以使用 CertOpenSystemStore 函数打开证书库，然后使用 CertEnum CertificatesInStore 函数枚举证书，那么在 CNG 中，是否有类似的函数进行证书操作呢？

实际上，在 CryptoAPI 中使用的证书操作接口同样适用于 CNG 环境，也可以说，证书库操作函数不是 CryptoAPI 库的专用函数，无论 CSP 还是 CNG 环境，都可以使用证书操作函数对证书进行存取操作。证书和密钥通过一些属性关联起来，这些关联属性因 CSP 或 CNG 而有些差别。

在 Windows 7 环境下，一个证书可能关联一个 CryptoAPI 密钥，也可能关联一个 CNG 密钥，如何判断证书关联的密钥类型呢？可以使用命令行工具 certutil 进行信息展示。使用如下命令列出当前系统证书库中安装的证书：certutil -user -repairstore my *。

此命令显示用户证书及其关联的私钥信息。如果证书有关联的私钥，此命令会显示私钥提供者是 CSP 或 CNG。如果是 CNG 提供者，则会显示“提供程序 = XXX Key Storage Provider”；否则就是 CSP 提供者。如下显示了 CNG 提供者证书和密钥。

```
序列号: 0128
颁发者: CN=OpenSSL CA, S=Beijing, C=CN
NotBefore: 2009-11-24 14:04
NotAfter: 2019-11-12 14:04
使用者: CN=ldapclient, S=Beijing, C=CN
非根证书
模板:
证书哈希(sha1): cf 65 1a ad 03 c4 a9 41 47 3a 7e 2a a1 19 3e de a5 00 a9 c8
密钥容器 = {40660087-6C63-4054-BD3C-982EAA77244D}
唯一容器名称:
 11cdae81ff4a19f0bc6e6c783f2be964_c40d85d9-e84f-4c76-b11b-df34193b54ed
提供程序 =  Microsoft Software Key Storage Provider
通过了加密测试
```

如果要查看系统安装的所有 CSP 和 CNG 提供者，可以使用命令“certutil -csplist”，如果“提供程序类型”信息为空，表示为 CNG 提供者，如下所示。

```
提供程序名称：Microsoft Enhanced Cryptographic Provider v1.0
提供程序类型：1 - PROV_RSA_FULL
提供程序名称：Microsoft Enhanced RSA and AES Cryptographic Provider
提供程序类型：24 - PROV_RSA_AES
提供程序名称：Microsoft Strong Cryptographic Provider
提供程序类型：1 - PROV_RSA_FULL
提供程序名称：Microsoft Software Key Storage Provider
提供程序名称：Microsoft Smart Card Key Storage Provider
```

前 3 个是 CSP 提供者，后 2 个是 CNG 提供者。注意，CNG 提供者名称中有“Key Storage Provider”。

在 Windows 7 环境下，导入 PKCS#12（.pfx 后缀）的证书私钥到微软的 CNG 密钥存储“Microsoft Software Key Storage Provider”提供者的命令行：

```
certutil -csp "Microsoft Software Key Storage Provider" -p 123456 -user -f -importPFX mycertkey.pfx
```

其中，“-csp”指定密钥提供者，此处使用微软 CNG 密钥存储容器，“-p”指定 mycertkey.pfx 文件的保护口令，“-user”为当前用户存储区，“-f”表示强制覆盖，如果已经存在对应证书私钥，则强制覆盖，“-importPFX”指定要导入的文件。

由于证书既可能关联 CSP 私钥，也可能关联 CNG 私钥，所以在程序中需要区分，以使用不同的密钥操作函数。为了获取证书关联的 CNG 私钥，同样需要使用 CryptAcquireCertificatePrivateKey 函数。在 Windows Vista 以后的版本中，微软扩展了此接口，使其可以支持获取 CNG 密钥。其函数原型如下：

```
BOOL WINAPI CryptAcquireCertificatePrivateKey(
__in    PCCERT_CONTEXT pCert,
__in    DWORD dwFlags,
__in    void *pvReserved,
__out HCRYPTPROV_OR_NCRYPT_KEY_HANDLE *phCryptProvOrNCryptKey,
__out   DWORD *pdwKeySpec,
__out   BOOL *pfCallerFreeProvOrNCryptKey);
```

在 dwFlags 中，可以指定如下标志，使 CryptAcquireCertificatePrivateKey 函数尝试 CNG 密钥：

① CRYPT_ACQUIRE_ALLOW_NCRYPT_KEY_FLAG：首先使用 CryptoAPI 方式获取私钥，如果失败，则使用 CNG 方式获取私钥。如果使用 CNG 方式获得私钥，则 pdwKeySpec 值会被设置为 CERT_NCRYPT_KEY_SPEC。

② CRYPT_ACQUIRE_ONLY_NCRYPT_KEY_FLAG：只使用 CNG 方式获取私钥，pdwKeySpec 中会被设置为 CERT_NCRYPT_KEY_SPEC。

由于证书使用方式与 CryptoAPI 相同，请参考 12.1 节的内容。

12.4.3 使用私钥

与 CSP 使用密钥方式类似，在使用密钥前，需要先获得密钥句柄，通过密钥句柄完成签名、加密等密钥运算。

在 12.1 节演示了如何获得 PCCERT_CONTEXT 类型的证书指针，下面介绍通过证书指针获得 CNG 密钥句柄，然后进行签名过程。

12.4.3.1 私钥签名过程

① 通过 CryptAcquireCertificatePrivateKey 获得私钥句柄。

② 如果关联私钥为 CNG 私钥，则调用 SignWithCngKey 函数进行签名。

③ 在 SignWithCngKey 函数中：

a. 首先使用 BCryptOpenAlgorithmProvider 获得哈希算法提供者，使用了默认提供者。

b. 通过 BCryptGetProperty 函数获得哈希对象内存大小。

c. 申请哈希对象内存空间。

d. 通过 BCryptGetProperty 函数获得哈希算法结果长度。

e. 申请哈希结果内存空间。

f. 通过 BCryptCreateHash 函数创建哈希对象。

g. 通过 BCryptHashData 函数计算数据的哈希值。

h. 通过 BCryptFinishHash 获得哈希结果。

i. 调用 NCryptSignHash 计算签名结果数据长度。

j. 申请签名结果空间。

k. 调用 NCryptSignHash 计算签名结果数据。

l. 清空申请的内存空间。

至此，在 pbSignature 中保存了签名结果，签名结果数据的长度为 cbSignature。

12.4.3.2　私钥签名示例程序

```
PCCERT_CONTEXT    pDesiredCert = ...;    // 已经通过证书库获得
HCRYPTPROV_OR_NCRYPT_KEY_HANDLE hcngKey;
DWORD dwSpec = 0;
BOOL    fCallerFreeProvOrNCryptKey;
BOOL rc = CryptAcquireCertificatePrivateKey(pDesiredCert,
      CRYPT_ACQUIRE_USE_PROV_INFO_FLAG|
      CRYPT_ACQUIRE_ALLOW_NCRYPT_KEY_FLAG,
      NULL,   &hcngKey,   &dwSpec,   &fCallerFreeProvOrNCryptKey);
if (!rc) {
      // CryptAcquireCertificatePrivateKey 打开私钥错误，可能证书没用关联私钥
      return;
}
if (dwSpec == CERT_NCRYPT_KEY_SPEC) {
      // 说明证书关联的是 CNG 类型私钥，调用 SignWithCngKey 函数进行签名
      SignWithCngKey((NCRYPT_KEY_HANDLE)hcngKey);
      printf("Cert key is CNG key. \n");
}

// 函数 SignWithCngKey 实现如下
#define KNOWN_DATA          "SignThis"
#define KNOWN_DATALEN       8
static void SignWithCngKey(NCRYPT_KEY_HANDLE hKey)
{
      BCRYPT_ALG_HANDLE                hAlgorithm          = NULL;
```