



证书管理器 Ver2.0 规格书

为方便用户管理本地私人证书集 (包括个人证书,他人证书,根证书以及 CRL 等), 也为软件开发人员调用与证书操作有关的且具有统一规格的公共接口函数, 使得此证书管理器有后继开发规范, 特制定此规格书.

此规格书只对软件现有功能作描述, 至于软件中数据库结构和软件流程将另附档案.

一 系统功能描述.....	3
二 API 函数说明	4
(一)内部证书处理模块.....	4
1. 添加用户证书和私钥 CMB_AddUserCertAndKey.....	4
2. 添加用户证书 CMB_AddUserCertificate	5
3. 取得用户证书对应的证书链 CMB_GetUserCertChain.....	6
4. 初始化数据库 CMB_InitialDataBase	6
5. 取得用户证书列表 CMB_GetUserCertificateLists.....	7
6. 关闭数据库 CMB_CloseDataBase	7
7. 获得证书详细内容 CMB_GetUserCertificate.....	8
8. 载入用户私钥至缓冲区 CMB_LoadUserPrivateKey	8
9. 删除个人证书 CMB_DeleteCertificate	9
10. 更新用户证书 CMB_UpdateUserCertificate.....	9
11. 更改用户证书密码 CMB_ChangePasswd.....	9
12. 导出用户证书 CMB_ExportUserCertificate	10
13. PKCS12 导出 CMB_p12CertExport	10
14. PKCS12 导入 CMB_p12CertImport	10
15. 清理内存私钥 CMB_ClearUserPrivateKey	10
16. 获取用户证书别名 CMB_GetCertViaName.....	11
17. 设置用户证书别名 CMB_SetCertViaName	11
18. 获取用户证书标识号 CMB_GetCertId.....	11
19. 获取用户证书 CMB_GetCertById.....	11
20. 获取用户证书状态 CMB_GetCertStatus.....	12
21. 设置用户证书状态 CMB_SetCertStatus	12
(二) 外部证书处理模块.....	13
1. 导出全部用户证书 CMB_ExportCertAll	13
2. 导入全部用户证书 CMB_ImportCertAll	13
3. 打开用 CMB_ExportCertAll () 导出的证书列表文件.....	13
4. 得到证书列表文件中下一个记录的信息	14
5. 插入用户证书到数据库.....	14
6. 插入用户证书和私钥到数据库	14
7. 关闭 3 打开的文件.....	15
三 加解密应用及证书编码模块.....	15
1. 获取证书信息 CMB_GetCertInfo.....	15
2. 获取证书细目 CMB_GetCertDetails.....	15
3. 证书 CRL 校验 CMB_VerifyCertByCRL	17
4. 证书 OCSP 校验 CMB_VerifyCertByOCSP	18



5. PEM 编码 CMB_PEMEncode.....	18
6. PEM 解码 CMB_PEMDecode.....	18
7. 签名 CMB_SignData.....	19
8. 验证签名 CMB_VerifySignData.....	19
9. 摘要 CMB_Digest.....	20
10. 数字信封 CMB_Envelope.....	20
11. 产生随机密钥 CMB_GenRandomBytes.....	20
12. 对称加密 CMB_EncryptData.....	21
13. 对称解密 CMB_DecryptData.....	21
四 错误处理函数.....	22
1. 得到错误消息.....	22

一 系统功能描述

以下是“Certificate Management Box”提供用户端个人证书(私钥)管理各功能,包括证书(私钥)的添加,更新,删除证书等。(证书,私钥一起存储或删除)

1. 证书列表树显示

列表树显示框内显示数据库中所有用户证书记录以及相应的证书链关系,每条记录包括证书UniqueID,证书持有人姓名,证书持有人邮件地址,证书状态(有效/无效),证书级别以及证书用途等常用证书信息.在此可以进行证书别名设置更改,匹配证书链等一系列证书相关功能的操作。

2. 证书导入

把证书(私钥)从证书原介质添加到本地硬盘,支持 p12 和 Sheca 证书格式.若证书已在本地,提示“不能添加证书”(直接导入方式)或“是否用新证书(私钥)替代旧证书(私钥)”(间接导入方式)。

3. 证书状态更新

由用户在证书列表树中选定一条记录后,选择菜单进行更新.此时可以根据用户所选择更新验证级别来进行证书的相应更新。

4. 证书删除

由用户在证书列表树选定一条记录后选择菜单进行记录删除.若存在对应私钥,连同私钥一齐删除。

5. 用户配置证书更新级别选项

证书链是否完整,是否处于有效期,是否处于CRL 废除列表,是否 OCSP废除。

6. 证书导出

从证书管理器中导出存在的用户证书,支持导出 Sheca 和 P12 证书格式.其中导出 p12 格式要求证书存在对应的私钥。

7. 修改私钥密码

若用户证书存在对应私钥,通过“密码修改”功能,可以进行用户私钥密码修改操作。

8. 证书服务

证书管理器中存在的所有证书,通过“证书服务”功能,可以链接到 SHECA 提供的相应证书服务URL,在使用过程中可以取得相应的在线帮助与服务。

二 API 函数说明

(一)内部证书处理模块

以下是“Certificate Management Box”提供程序员端个人证书(密钥)功能函数,由动态连接库(ShecaCertMgr.dll)提供.

1. 添加用户证书和私钥 CMB_AddUserCertAndKey

```
int CMB_AddCertAndKey(unsigned short certificatedevicetype,
                      char *certificatedeviceparameter,
                      char *certificatepassword,
                      unsigned short privatekeydevicetype,
                      char *privatekeydeviceparameter,
                      char *privatekeypassword,
                      int ParentCertId,
                      int *CertMgrCertId,
                      char *CertCommonName)
```

● 存储新证书和对应私钥至证书管理器

参数名	含义	输入/ 输出	参数选项
Certificatedevicetype	存储证书的设备类型	In	1=文件, 其他见设备类型编码表
Certificatedevice Parameter	存储私钥设备的参数	In	1,为文件名,其他可为”com1”, “com2”
Certificatepassword	证书密码	In	
Privatekeydevicetype	存储私钥的设备类型	In	
Privatekeydevice Parameter	存储私钥设备的参数	In	1,为文件名,其他可为”com1”, “com2”
Privatekeypassword	私钥密码	In	
ParentCertId	上一级父证书的标识号	In	
CertMgrCertId	证书管理器证书标识号	Out	注意: 对于用户证书, 证书管理器证书标识号 = 用户证书标识号 (CertId) + 10000; 根级证书, 证书管理器证书标识号 = 用户证书标识号(CertId);

CertCommonName	用户证书通用名称	Out	
----------------	----------	-----	--

返回:

0	成功
CMB_ERROR_READFILE	读文件错误
CMB_ERROR_READIC	读IC卡错误
CMB_ERROR_PASSWORD	私钥口令错误
CMB_ERROR_KEYCERT_PAIR	私钥与证书不匹配
CMB_ERROR_UNSUPPORTED_DRIVER	IC卡或读卡器不支持
CMB_ERROR_DRIVER_VERSION	驱动程序版本错误
CMB_ERROR_FORMAT	证书格式错误
CMB_ERROR_DLL	动态连接库错误
CMB_ERROR_DATABASE	数据库错误

设备类型编码表

ICDriver	ICCard	文件 (1)	Sheca (2)	社保卡 (3)	代码卡(4)	EKey(5)	缓冲区(6)
长丰读卡器	(1)	0x0001	0x0102	0x0103	0x0104	0x0005	0x0006
ChipDriver	(2)	0x0001	0x0202	0x0203	0x0204	0x0005	0x0006
麦柯	(3)	0x0001	0x0302	0x0303	0x0304	0x0005	0x0006
Gemplus	(4)	0x0001	0x0402	0x0403	0x0404	0x0005	0x0006
SHECA 读卡器	(5)	0x0001	0x0502	0x0503	0x0504	0x0005	0x0006

2. 添加用户证书 CMB_AddUserCertificate

```
int CMB_AddUserCertificate(unsigned short certificatedevicetype,
                           char *certificatedeviceparameter,
                           char *certificatepassword,
                           unsigned short privatekeydevicetype,
                           char *privatekeydeviceparameter,
                           char *privatekeypassword,
                           int ParentCertId,
                           int * CertMgrCertId,
                           char * CertCommonName) ;
```

● 存储新证书至证书管理器

参数名	含义	输入 / 输出	参数选项
Certificatedevicetype	存储证书的设备类型	In	1=文件, 6=缓冲区, 其他见设备类型编码表
CertificatedeviceParameter	存储私钥设备的参数	In	1,为文件名, 6,为内存缓冲区, 其他可为"com1", "com2"
Certificatepassword	证书密码	In	6,该参数为空。
Privatekeydevicetype	存储私钥的设备类型	In	1=文件, 其他见设备类型编码表

Privatekeydevice Parameter	存储私钥设备的 参数	In	1,为文件名, 其他可为"com1", "com2"
Privatekeypassword	私钥密码	In	
ParentCertId	上一级父证书 的标识号	In	
CertMgrCertId	证书管理器证 书标识号	Out	注意: 对于用户证书,证书管理器证书 标识号 = 用户证书标识号 (CertId) + 10000; 根级证书,证书管理器证书标识 号 = 用户证书标识号(CertId);
CertCommonName	用户证书通用 名称	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

3. 取得用户证书对应的证书链 CMB_GetUserCertChain

```
long CMB_GetUserCertChain(unsigned short certificatedevicetype,
                           char* certificatedeviceparameter,
                           char* CertChainCetIdList,
                           char* CertChainNameList,
                           int* CertChainNums);
```

● 取得用户证书对应的证书链

参数名	含义	输入 / 输出	参数选项
Certificatedevicetype	存储证书的设备 类型	In	1=文件, 6=缓冲区, 其他见设备类型编码表
Certificatedevice Parameter	存储私钥设备的 参数	In	1,为文件名, 6, 为内存缓冲区, 其他可为"com1", "com2"
Certificatepassword	密码	In	6, 该参数为空。
CertChainCertIdList	证书链中根证书 证书标识列表	Out	
CertChainNameList	证书链中根证书 通用名列表	Out	
CertChainNums	证书链中根证书 的数目	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

4. 初始化数据库 CMB_InitialDataBase

```
int CMB_InitialDataBase(STRUCT dbcontext* context)
```

● 初始化数据库

参数名	输入 / 输出	含义
Context	Input/Output	数据库信息

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

5. 取得用户证书列表 CMB_GetUserCertificateLists

```
int CMB_GetUserCertificateLists(STRUCT dbcontext* db,
                               int *CertMgrCertId,
                               char *commonName, char *ViaName,
                               char *CertSerialNo,
                               char* status, char* PEMCert,
                               char* CertChainCertIdList,
                               char* CertChainNameList,
                               int* CertChainNums);
```

● 循环调用取得用户证书数据库中证书列表.

参数名	输入 / 输出	含义
Db	In	数据库结构
CertMgrCertId	Out	证书管理器证书标识号
CommonName	Out	证书通用名
ViaName	Out	证书别名
CertSerialNo	Out	证书系列号
Status	Out	证书状态字段(暂定为8个字符, 每个字符以‘0’表示有效, ‘1’表示无效。其中: Char[0]:Status字段的有效性, char[1]:证书链完整性, char[2], 证书有效期, char[3], crl 校验值, char[4], ocsps校验值, char[5], 有无私钥)
PEMCert	Out	PEM编码证书
CertChainCertIdList	Out	证书链中根证书序列号列表
CertChainNameList	Out	证书链中根证书通用名列表
CertChainNums	Out	证书链中根证书的数目

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

6. 关闭数据库 CMB_CloseDataBase

```
int CMB_CloseDataBase(STRUCT dbcontext* DataBase)
```

● 关闭数据库

参数名	输入 / 输出	含义
DataBase	In	数据库结构

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

7. 获得证书详细内容 CMB_GetUserCertificate

```
int CMB_GetUserCertificate(char* CertId,
                          int CertType,
                          char* CommonName,
                          char* ViaName,
                          char * SerialNo;
                          char* Status,
                          char* PEMCert);
```

- 按证书 UniqueID 从数据库中获得证书详细内容

参数名	含义	输入 / 输出	参数选项
CertId	证书标识号	In	
CertType	证书类型	In	0,用户证书, 1, 根证书
CommonName	证书通用名	Out	
ViaName	证书别名	Out	
SerialNo	证书系列号	Out	
Status	证书状态	Out	同上
PEMCert	PEM编码证书	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

8. 载入用户私钥至缓冲区 CMB_LoadUserPrivateKey

```
int CMB_LoadUserPrivateKey(char * CertId, char *privatekeypassword);
```

- 载入所选私钥

参数名	含义	输入 / 输出	参数选项
CertId	用户证书标识号	In	
Privatekeypassword	私钥密码	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

9. 删除个人证书 CMB_DeleteCertificate

```
int CMB_DeleteCertificate(char *CertId[input],  
                          int CertType[input]);
```

10. 更新用户证书 CMB_UpdateUserCertificate

```
int CMB_UpdateUserCertificate(int CertId,  
                              int CertType,  
                              int CertCheckMode,  
                              char* CertStatus);
```

● 更新证书状态

参数名	输入 / 输出	含义
CertId	In	证书标识号
CertType	In	证书类型
CertCheckMode	In	证书校验模式
CertStatus	Out	更新后的证书状态

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

11. 更改用户证书密码 CMB_ChangePasswd

```
int CMB_ChangePasswd( char *CertId, char *OldPassword, char *NewPassword)
```

功能: 更改存储介质的密码。

参数:

参数名	含义	输入 / 输出	参数选项
CertId	证书标识号	In	
OldPassword	旧密码	In	
Newpassword	新密码	In	

返回:

0	正常返回,密码更改成功
CMB_ERROR_READFILE	读写文件错
CMB_ERROR_READIC	读写 IC 卡出错
CMB_ERROR_PASSWORD	口令检查不通过
CMB_ERROE_CRYPT	加解密模块出错
CMB_ERROR_DATABASE	数据库错误
CMB_ERROR_DLL	动态库出错
CMB_ERROR_UNSUPPORTED_DRIVER	不支持的 IC 卡或读卡器
CMB_ERROR_DRIVER_VERSION	IC卡驱动程序版本错误

12. 导出用户证书 **CMB_ExportUserCertificate**

```
int CMB_ExportUserCertificate(char *CertId,  
                             int CertType,  
                             char *filename)
```

- 导出用户证书

参数名	输入 / 输出	含义
CertId	In	证书标识号
CertType	In	证书类型
FileName	In	导出证书的文件名

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

13. PKCS12 导出 **CMB_p12CertExport**

```
int CMB_p12CertExport(int CertId,  
                      char* Filename,  
                      char* password);
```

参数名	输入 / 输出	含义
CertId	In	证书标识号
Filename	In	导出文件名
Password	In	私钥密码

14. PKCS12 导入 **CMB_p12CertImport**

```
CMB_p12CertImport(char* filepath[input],char*password[input],  
                  int *CertId[output],  
                  char * username[output],  
                  char * CertPCertIdList[output],  
                  char * CertPNameList[output],  
                  int * CertPNum[output]);
```

15. 清理内存私钥 **CMB_ClearUserPrivateKey**

```
int CMB_ClearUserPrivateKey();
```

16. 获取用户证书别名 CMB_GetCertViaName

```
int CMB_GetCertViaName(char *CertId, char * ViaName);
```

- 载入所选私钥

参数名	含义	输入 / 输出	参数选项
CertId	证书标识号	In	
ViaName	证书别名	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

17. 设置用户证书别名 CMB_SetCertViaName

```
int SetCertViaName (char * CertId[input], char * ViaName[input]);
```

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

18. 获取用户证书标识号 CMB_GetCertId

```
int CMB_GetCertId(char* PemCert,  
                 int CertType,  
                 int * CertId,  
                 int * ParentCertId);
```

- 通过用户证书获取数据库标识号

参数名	含义	输入 / 输出	参数选项
PemCert	Pem 码证书	In	
CertType	证书类型	In	
CertId	证书标识号	Out	
ParentCertId	父证书标识号	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

19. 获取用户证书 CMB_GetCertById

```
int CMB_GetCertById(char * CertId,  
                   int CertType,
```

```
char * PemCert);
```

● 通过标识号获取证书

参数名	含义	输入 / 输出	参数选项
CertId	证书标识号	In	
CertType	证书类型	In	
PemCert	Pem 编码证书	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

20. 获取用户证书状态 CMB_GetCertStatus

```
int CMB_GetCertStatus(int CertId,  
                      int CertType,  
                      char * VerifyClass  
                      char *CertStatus);
```

● 通过标识号获取证书

参数名	含义	输入 / 输出	参数选项
CertId	证书标识号	In	
CertType	证书类型	In	
VerifyClass	证书状态的验证级别	Out	0, 仅证书链, 有效期校验, 1, 仅 0&CRL 校验, 2, 仅 0& OCSP 校验, 3, 仅 0&CRL&OCSP 校验
CertStatus	证书状态	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

21. 设置用户证书状态 CMB_SetCertStatus

```
int CMB_SetCertStatus(char * CertId,  
                      int CertType,  
                      char * VerifyClass,  
                      char * CertStatus);
```

● 通过标识号获取证书

参数名	含义	输入 / 输出	参数选项
CertId	证书标识号	In	
CertType	证书类型	In	
VerifyClass	证书状态的验证级别	In	同上

CertStatus	证书状态	In	
------------	------	----	--

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

(二) 外部证书处理模块

1. 导出全部用户证书 CMB_ExportCertAll

```
int CMB_ExportCertAll (char* filepath[input],char* password[input]);
```

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

2. 导入全部用户证书 CMB_ImportCertAll

```
int CMB_ImportAll (char *filepath[input], char* password[input],bool overwrite[input]);
```

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

3. 打开用 CMB_ExportCertAll () 导出的证书列表文件

```
int CMB_OpenExportedFile(char *filepath[input], char* password[input]);
```

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

4. 得到证书列表文件中下一个记录的信息

```
int CMB_GetNextExportedCertificate(char* CertSN  
                                char* CommonName,  
                                char* Email,  
                                char* PemCert,  
                                char* PemKey,  
                                char* KeyMedium,  
                                char* KeyPath);
```

- 得到证书列表文件中下一个证书记录的信息

参数名	输入 / 输出	含义
CertSN	Out	证书序列号
CommonName	Out	证书通用名
Email	Out	Email
PemCert	Out	PEM编码证书
PemKey	Out	PEM编码私钥
KeyMedium	Out	私钥保存介质
KeyPath	Out	私钥保存路径

返回值:

0	成功
CMB_ERROR_NOMORE_RECORD	已到文件尾
<0	用CMB_Get_Error_Message() 函数获取错误信息

5. 插入用户证书到数据库

```
int CMB_InsertCertificate(char* certpem[input],bool overwrite[input]);
```

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

6. 插入用户证书和私钥到数据库

```
int CMB_InsertCertAndKey(char * CertSN[input],  
                          char * CommonName[input],  
                          char * Email[input],  
                          char* PemCert[input],  
                          char* PemKey[input],  
                          char* Keymedium[input],  
                          char* Keypath[input]);
```

```
bool overwrite[input]);
```

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

7. 关闭 3 打开的文件

```
int CMB_CloseExportedFile();
```

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

三 加解密应用及证书编码模块

1. 获取证书信息 **CMB_GetCertInfo**

```
int CMB_GetCertInfo(unsigned char *PEMCert, unsigned short PEMCertLen,  
CERT_INFO *pCertInfo);
```

● 取证书的信息

参数名	含义	输入 / 输出	参数选项
PEMCert	待解码的证书的起始地址	In	
PEMCertLen	待解码的证书的的长度	In	
PcertInfo	解码后的证书结构的指针	In, Out	

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

2. 获取证书细目 **CMB_GetCertDetails**

```
int CMB_GetCertDetails(char* PemCert,  
unsigned short ItemNo,  
unsigned char *ItemValue,  
unsigned short *ItemLength)
```

● 取证书的细目

参数名	含义	输入 / 输出	参数选项
PEMCert	待解码的证书的起始地址	In	

ItemNo	项目 id	In	1 证书版本 2 证书序列号 3 证书签名算法 4 证书发放者国家名 5 证书发放者组织名 6 证书发放者部门名 7 证书发放者省州名 8 证书发放者通用名 9 证书发放者城市名 10 证书发放者EMAIL地址 11 证书有效期起始 12 证书有效期截止 13 用户国家名 14 用户组织名 15 用户部门名 16 用户省州名 17 用户通用名 18 用户城市名 19 用户EMAIL地址 20 用户DER公钥值 21 用户证书自定义级别 22 用户证书用途 23 用户证书服务URL
ItemValue	解码后的项目指针	Out	由外部分配
Itemlength	项目长度	In/Out	输入缓冲区长度, 返回解码后项目长度

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

函数返回值说明:

用户证书自定义级别 : 字符串形式表示的整形值. 其具体含义如下:

0: "没有扩展项"
0x00000103: "UC 个人身份证书"
0x00000102: "UC 个人 EMAIL 证书"
0x00000105: "UC 个人代码签名证书"
0x00000203: "UC 单位身份证书"
0x00000202: "UC 单位 EAIL 证书"
0x00000204: "UC 单位代码签名证书"
0x00000304: "UC 服务器身份证书"
0x00000301: "UCWEB 服务器证书"
0x00000104: "SET 个人证书"
0x00000205: "SET 单位证书");
0x00000302: "SET 商户证书"

用户证书用途 : 字符串形式表示的整形值. 其异或结果具体含义如下:

```
#define KEYUSAGE_digitalSignature    0x0001
#define KEYUSAGE_nonRepudiation      0x0002
#define KEYUSAGE_keyEncipherment     0x0004
#define KEYUSAGE_dataEncipherment    0x0008
#define KEYUSAGE_keyAgreement        0x0010
#define KEYUSAGE_keyCertSign         0x0020
#define KEYUSAGE_crlSign              0x0040
#define KEYUSAGE_encipherOnly        0x0080
#define KEYUSAGE_decipherOnly        0x0100
```

3. 证书 CRL 校验 **CMB_VerifyCertByCRL**

```
int CMB_VerifyCertByCRL(unsigned char *DerCert,
                        unsigned short DerCertLen,
                        int isRoot);
```

● 证书 CRL 校验

参数名	含义	输入 / 输出	参数选项
DerCert	Der 码证书	In	
DerCertLen	证书长度	In	
IsRoot	根证书标志	In	0,用户证书 1, 根证书

返回值:

0	CRL校验通过
其它	用CMB_Get_Error_Message () 函数获取错误信息

4. 证书 OCSP 校验 **CMB_VerifyCertByOCSP**

```
int CMB_VerifyCertByOCSP (unsigned char *DerCert,  
                          unsigned short DerCertLen,  
                          unsigned char *ParentDerCert,  
                          unsigned short ParentDerCertLen,  
                          int isRoot)
```

● 证书 OCSP 校验

参数名	含义	输入 / 输出	参数选项
DerCert	Der 码证书	In	
DerCertLen	证书长度	In	
ParentDerCert	DER 编码父证书	In	
ParentDerCertLen	父证书长度	In	
IsRoot	根证书标志	In	0,用户证书 1, 根证书

返回值:

0	OCSP校验通过
<0	用CMB_Get_Error_Message () 函数获取错误信息

5. PEM 编码 **CMB_PEMEncode**

```
int CMB_PEMEncode(unsigned char *indata[input], unsigned long indatalen[input],  
                  unsigned char *pemdata[output], unsigned long *pemlen[output])
```

● PEM编码

参数名	含义
Indata	待编码的数据
Indatalen	待编码数据的长度
Pemdata	Pem 编码后的数据
Pemlen	Pem 编码数据的长度

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

6. PEM 解码 **CMB_PEMDecode**

```
int CMB_PEMDecode(unsigned char *pemdata[input], unsigned long pemlen[input],  
                  unsigned char *outdata[output], unsigned long *outlen[output])
```

● PEM解码

参数名	含义
Pemdata	待解码的数据
Pemlen	待解码数据的长度

Outdata	解码后的数据
Outlen	解码数据的长度

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

7. 签名 CMB_SignData

```
int CMB_SignData (unsigned char *origindata[input], unsigned long length[input],  
                 unsigned short signmethod[input], unsigned char *signeddata[output],  
                 unsigned long *signedlength[output])
```

● 数据签名

参数名	含义	参数选项
Origindata	原始数据块	
Length	原始数据库长度	
Signmethod	签名类型	1: MD2 2: MD5 3: SHA1
Signeddata	处理后数据块	
Signedlength	处理后数据库长度	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

8. 验证签名 CMB_VerifySignData

```
int CMB_VerifySignData(unsigned char *origindata, unsigned long originlength,  
                       unsigned short signmethod, unsigned char *signeddata,  
                       unsigned long signedlength, unsigned char *certificate,  
                       unsigned short certlength)
```

● 验证签名

参数名	含义	In/out	参数选项
Origindata	原始数据块	In	
Originlength	原始数据库长度	In	
Signmethod	签名类型	In	1: MD2 2: MD5 3: SHA1
Signeddata	签名数据块	In	
Signedlength	签名数据块长度	In	
Certificate	证书内容	In	
Certlength	证书长度	In	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

9. 摘要 CMB_Digest

```
int CMB_Digest(unsigned char *data, unsigned long length, unsigned short method,  
              unsigned char *digest, unsigned long *digestlength)
```

● 摘要

参数名	含义	In/out	参数选项
Data	原始数据块	In	
Length	原始数据库长度	In	
Method	摘要方法	In	1: MD2 2: MD5 3: SHA1
Digest	处理后数据块	Out	
Digestlength	处理后数据长度	Out	

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

10. 数字信封 CMB_Envelope

```
int CMB_Envelope(unsigned short envelopetype, unsigned char *indata,  
                unsigned long inlength, unsigned char *outdata, unsigned long *outlength,  
                unsigned char *PEMcertificate, unsigned short certlength)
```

● 数字信封打包或拆解

参数名	含义	In/out	参数选项
Envelopetype	信封类型	In	1 组成数字信封 2 拆解数字信封
Indata	原始数据块	In	
Inlength	原始数据库长度	In	
Outdata	处理后数据块	Out	由外部分配
Outlength	处理后数据库长度	Out	
PEMcertificate	证书内容	In	在解数字信封时不起作用
Certlength	证书长度	In	在解数字信封时不起作用

返回:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

11. 产生随机密钥 CMB_GenRandomBytes

```
int CMB_GenRandomBytes(unsigned char *randombytes, int randombyteslen)
```

● 产生随机密钥

参数名	含义	输入 / 输出	参数选项
Randombytes	产生的随机数数据	Out	

Randombyteslen	要产生的随机数长度	In	
----------------	-----------	----	--

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

12. 对称加密 CMB_EncryptData

```
int CMB_EncryptData(unsigned char *indata, int indatalen, unsigned char *outdata,  
                    int *outdatalen, unsigned char *sdbikey)
```

- 用对称算法 (SDBI) 加密应用数据

参数:

参数名	含义	输入 / 输出	参数选项
Indata	要加密的数据	In	
Indatalen	要加密的数据长度	In	
Outdata	加密后的数据	Out	
Outdatalen	加密后的数据长度	Out	
Sdbikey	SDBI 算法加密的对称密钥 (长度为 16 字节)	In	

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

13. 对称解密 CMB_DecryptData

```
int CMB_DecryptData(unsigned char *indata, int indatalen, unsigned char *outdata,  
                    int *outdatalen, unsigned char *sdbikey)
```

- 用对称算法 (SDBI) 解密应用数据

参数名	含义	输入 / 输出	参数选项
Indata	要解密的数据	In	
Indatalen	要解密的数据长度	In	
Outdata	解密后的数据	Out	
Outdatalen	解密后的数据长度	Out	
Sdbikey	解密的对称密钥	In	

返回值:

0	成功
<0	用CMB_Get_Error_Message () 函数获取错误信息

四 错误处理函数

1、得到错误消息

```
int CMB_Get_Error_Message(int err_code, char* err_msg);
```

参数名	含义	输入 / 输出	参数选项
Err_code	错误返回码	In	
Err_msg	错误信息	output	Char[256]