

## Криптографическая система ШИФР-РКІ

Книга 2

Базовый комплекс системы Шифр-РКІ

Часть 5

Модуль криптографических функций

Руководство программисту

Киев  
2004

---

**Copyright © 2002-2004 ЗАО «Сайфер»**

Описываемый в настоящей книге программный продукт и связанная с ним документация защищаются от несанкционированного копирования, распространения и использования в соответствии с законодательством Украины и международными соглашениями.

Описываемый программный продукт не может быть частично и полностью реализован в любой форме или любым способом без предварительного письменного разрешения ЗАО «Сайфер».

---

**ЗАО «Сайфер»**

Киев, ул. Нагорная д.25

Тел., факс (044) 246-99-00, 246-98-35, 213-03-22

E-mail: [sales@cipher.kiev.ua](mailto:sales@cipher.kiev.ua)

<http://www.cipher.kiev.ua>

## Оглавление

Оглавление.....	3
Общая характеристика Модуля криптографических функций.....	5
Назначение.....	5
Состав программных средств.....	6
Требования к программному и аппаратному обеспечению.....	6
Описание модуля криптографических функций.....	7
Функции инициализации и вспомогательные функции.....	7
SLGetStatusText.....	7
SLInitializeLibrary.....	7
SLDeinitializeLibrary.....	7
SLGetLibraryVersion.....	8
SLCreateCtx.....	8
SLCreateCtxEx.....	8
SLDuplicateCtx.....	9
SLDestroyCtx.....	9
SLInitialize.....	9
SL_LOG_CALLBACK.....	10
SLSetLogCallback.....	10
SLGetLogCallback.....	11
Функции для работы с ключами.....	12
SLLoadPrivateKey.....	12
SLNextEnumKeyEx.....	13
SLEndEnumKeys.....	13
Функции выработки и проверки цифровой подписи.....	15
SLGetSignatureSize.....	15
SLBeginHashing.....	15
SLUpdateFileHash.....	15
SLUpdateMemoryHash.....	16
SLProduceSign.....	16
SLVerifySign.....	16
Функции шифрования данных.....	17
SLGetEncryptionHdrSizeEx.....	17
SLSetDefaultKeyIDType.....	17
SLEncryptFileEx.....	17
SLDecryptFileEx.....	18
SLEncryptDataEx.....	19
SLDecryptDataEx.....	19
SL_READ_CALLBACK.....	20
SL_WRITE_CALLBACK.....	20
SLEncryptDataCB.....	20
SLDecryptDataCB.....	21
Структуры.....	23
CSLSignInfo.....	23
CSLKeyInfoEx.....	23
CSLKeyEnumInfoEx.....	24
Профили (типы) ключей.....	25
Флаги, описывающие полномочия ключа.....	25
Значения, описывающие состояние ключа.....	26
Описание слова статуса.....	27

---

---

Формат слова статуса.....	27
Макросы для проверки кода критичности.....	27
Макросы для выделения полей статуса.....	27

## Общая характеристика Модуля криптографических функций

### Назначение

*Модуль криптографических функций* является исполнительным устройством криптографической системы Шифр-РКІ и предназначен для криптографической защиты данных (имитозащита, шифрование, цифровая подпись) в распределенных автоматизированных системах и комплексах различного назначения.

Основными функциями *Модуля* являются следующие:

- выработка хэш-функции файла (участка файла)
- выработка хэш-функции блока данных (в памяти)
- постановка и проверка электронной цифровой подписи файла (участка файла)
- постановка и проверка электронной цифровой подписи блока данных (в памяти)
- шифрование и расшифрование данных файла (участка файла)
- шифрование и расшифрование блока данных (в памяти)
- поиск в таблице сертификата ключа по заданным критериям и выдачу информации о найденном сертификате
- выдачу информации о выполненных действиях для протоколирования в системном журнале.

При функционировании *Модуля* обеспечивается:

- ввод секретного ключа, его расшифрование и сохранение в памяти на время работы Модуля
- выдача информации о загруженном секретном ключе
- остановка работы Модуля при несоответствии введенного секретного ключа соответствующему сертификату, либо если использование соответствующего сертификата заблокировано
- поиск в таблице и выбор сертификата, требуемого для выполнения криптопреобразований:
  - по ID владельца сертификата (идентификатор системы Шифр – РКІ)
  - по ID ключа (идентификатор системы Шифр – РКІ)
  - по имени владельца сертификата
  - по электронному адресу владельца сертификата
  - по внешне задаваемому числовому идентификатору
  - по внешне задаваемому текстовому идентификатору
- проверка аутентичности выбранного сертификата.

В дополнении к перечисленным действиям при выполнении функций шифрования и расшифрования *Модуль* обеспечивает:

- выработку ключей связи по протоколу Диффи-Хеллмана
- генерацию ключей шифрования данных
- выработку имитовставки перед выполнением процедуры шифрования
- контроль целостности данных путем проверки имитовставки после выполнения процедуры расшифрования.

---

## Состав программных средств

---

*Модуль криптографических функций* представляет собой унифицированные и универсальные средства криптографической защиты информации, легко адаптируемые для использования в автоматизированных системах и комплексах различного функционального назначения.

*Модуль криптографических функций* поставляется в виде динамически подключаемых библиотек (в виде DLL для Win 32). В комплект поставки входят:

1. **c32csp.dll** – библиотека криптографических процедур (программное изделие «Шифр»);
2. **seclib.dll** - библиотека функций цифровой подписи и шифрования файлов;
3. **ncertAPI.dll, stbKeys.dll** - библиотеки функций управления ключами и сертификатами;
4. **CPKICServer.dll** – библиотека функций для работы с сервером сертификатов
5. **borlndmm.dll, stbRtl.dll** – служебные библиотеки

В комплект поставки также входят:

1. Файл прототипов процедур модуля криптографических функций – **seclib.h**.
2. Файл описаний статусов возврата модуля криптографических функций – **SLStatus.h**.
3. Файл описаний статусов возврата модуля управления ключами - **CSStatus.h**.
4. Файл импорта функций для работы в среде Visual C++ 6.0 - **SecLib.lib**.
5. Примеры применения функций библиотеки **seclib.dll** (в исходных текстах).

---

## Требования к программному и аппаратному обеспечению

---

*Модуль криптографических функций* предназначен для работы под управлением операционных систем Win9X/Me/NT/2000/XP в компьютерах с процессором не ниже Pentium или совместимых с ним.

## Описание модуля криптографических функций

### Функции инициализации и вспомогательные функции.

#### **SLGetStatusText**

Возвращает текстовое описание статуса. Строка формируется во внутреннем статическом буфере, поэтому последующие вызовы функции перезапишут её. Рекомендуется сразу же после получения строки скопировать её в свой буфер.

```
SECLIB_API(const char*) SLGetStatusText( SLSTATUS Status );
```

#### Параметры:

Параметр	Назначение
Status	Статус, текстовое описание которого необходимо получить.

#### Возвращает:

Указатель на текстовую строку с описанием статуса. Строка завершается нулевым символом.

#### **SLInitializeLibrary**

Выполняет инициализацию библиотеки. Эта функция должна быть вызвана перед вызовом любой другой функции библиотеки один раз в пределах одного процесса.

```
SECLIB_API(SLSTATUS)  
SLInitializeLibrary(void);
```

#### Параметры:

нет

#### Возвращает:

Статус

#### **SLDeinitializeLibrary**

Выполняет деинициализацию библиотеки.

```
SECLIB_API(SLSTATUS)  
SLDeinitializeLibrary(void);
```

#### Параметры:

нет

#### Возвращает:

Статус

## SLGetLibraryVersion

Возвращает информацию о версии библиотеки. Нужная библиотека указывается параметром nLibType. Версия возвращается в следующем

```
SECLIB_API(unsigned long) SLGetLibraryVersion( unsigned nLibType );
```

### Параметры:

Параметр	Назначение
nLibType	Тип библиотеки, версия которой запрашивается. Допустимы следующие значения параметра: VERSION_INFO_SECLIB - библиотека SecLib VERSION_INFO_CERTSAPI – библиотека управления ключами.

### Возвращает:

Версию библиотеки как 32-битовое число в следующем формате:

Биты	Назначение
0-15	Номер редакции библиотеки (1.01. <b>0001</b> )
16-23	Младший номер версии (1. <b>01</b> .0001)
24-31	Старший номер версии (1.01. <b>0001</b> )

## SLCreateCtx

Создаёт контекст библиотеки. Все операции с данными требуют передачи созданного этой функцией дескриптора контекста.

**\*\*ВНИМАНИЕ\*\***

Контекст не является реентерабельным (один и тот же контекст нельзя использовать одновременно в разных потоках).

```
SECLIB_API(SLSTATUS)
SLCreateCtx( HSECLIB* phCtx );
```

### Параметры:

Параметр	Назначение
phCtx	Указатель на переменную, в которую будет занесён дескриптор созданного контекста

### Возвращает:

Статус

## SLCreateCtxEx

Создаёт контекст библиотеки. Все операции с данными требуют передачи созданного этой функцией дескриптора контекста.

**\*\*ВНИМАНИЕ\*\***

Контекст не является реентерабельным (один и тот же контекст нельзя использовать одновременно в разных потоках).

```
SECLIB_API(SLSTATUS)
SLCreateCtxEx( HSECLIB* phCtx, unsigned nMode );
```

**Параметры:**

Параметр	Назначение
phCtx	Указатель на переменную, в которую будет занесён дескриптор созданного контекста
nMode	режим работы библиотеки. Возможны следующие значения: SL_MODE_DEFAULT – нормальный режим работы, SL_MODE_REMOTECLIENT - режим работы удалённого клиента (например юридического или физического лица) SL_MODE_TCPCLIENT – работа с сервером сертификатов

**Возвращает:**

Статус

**SLDuplicateCtx**

Создаёт копию контекста библиотеки.

```
SECLIB_API(SLSTATUS)
SLCreateCtxEx( HSECLIB* phCtx, unsigned nMode );
```

**Параметры:**

Параметр	Назначение
phDestCtx	Указатель на переменную, в которую будет занесён дескриптор копии контекста
hSrcCtx	Дескриптор копируемого контекста.

**Возвращает:**

Статус

**SLDestroyCtx**

Уничтожает контекст библиотеки.

```
SECLIB_API(SLSTATUS)
SLDestroyCtx( HSECLIB hCtx );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор уничтожаемого контекста.

**Возвращает:**

Статус

**SLInitialize**

Инициализирует контекст библиотеки.

```
SECLIB_API(SLSTATUS)
SLInitialize( HSECLIB hCtx, const char* cszCfgDir );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор инициализируемого контекста.
cszCfgDir	Полный путь к каталогу с ключевой информацией Если при создании контекста был указан режим <b>SL_MODE_TCPCLIENT</b> , то в данный параметр должен содержать TCP адрес сервера сертификатов в следующем формате: IP-адрес:порт или доменное имя:порт. Если порт не указан, то используется значение по умолчанию 831. Например: 192.168.0.1:831 localhost:831 cert.server.ua:831

**Возвращает:**

Статус

**SL\_LOG\_CALLBACK**

Прототип функции обратного вызова для протоколирования сообщений библиотеки.

```
typedef int (__stdcall *SL_LOG_CALLBACK)( void* pCustomData, SLSTATUS Status, const char* cszLogText );
```

**Параметры:**

Параметр	Назначение
pCustomData	Значение пользовательского указателя, переданного при вызове функции SLSetCallback
Status	Статус завершения операции, вызвавшей данное сообщение.
cszLogText	Текстовая строка сообщения, заканчивающаяся нулевым символом.

**Возвращает:**

Статус

**SLSetLogCallback**

Устанавливает функцию обратного вызова, которая будет вызываться каждый раз, когда библиотеке необходимо запротоколировать сообщение.

```
SECLIB_API(SLSTATUS)
SLSetLogCallback( HSECLIB hCtx, SL_LOG_CALLBACK pfnCallback, void* pCustomData );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
pfnCallback	Указатель на функцию обратного вызова, значение NULL запрещает вызов данной функции.

pCustomData	Произвольный указатель, определяемый пользователем, значение которого будет передаваться функции обратного вызова.
-------------	--

**Возвращает:**

Статус

**SLGetLogCallback**

Возвращает текущее значение указателя на функцию обратного вызова для протоколирования сообщений библиотеки. Предназначена для реализации протоколирования работы библиотеки.

```
SECLIB_API(SLSTATUS)
SLGetLogCallback( HSECLIB hCtx, SL_LOG_CALLBACK* ppfnCallback,
void** ppCustomData );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
ppfnCallback	Указатель для возврата текущего значения указателя на функцию обратного вызова.
ppCustomData	Указатель на переменную для возврата пользовательского указателя. Если указано NULL, то данный параметр игнорируется.

**Возвращает:**

Статус

## Функции для работы с ключами

### SLLoadPrivateKey

Загружает секретный ключ в память и выдаёт информацию о нём. После успешной загрузки становятся доступными функции шифрования и цифровой подписи.

SECLIB\_API(SLSTATUS)

SLLoadPrivateKey( HSECLIB hCtx, const char\* cszKeyDir,  
const char\* cszKeyPassword, struct CSLKeyInfo\* pKeyInfo )

#### Параметры:

Параметр	Назначение
hCtx	<b>Дескриптор контекста.</b>
cszKeyDir	Полный путь к каталогу с секретным ключом. Для чтения ключей из Touch Memory необходимо вместо пути передать следующую строку: "@". Предварительно должна быть установлена и конфигурирована поддержка Touch Memory. При использовании в качестве ключевого носителя смарт-карт необходимо указать следующую строку: "#"
cszKeyPassword	Пароль секретного ключа
pKeyInfo	Указатель на структуру типа CSLKeyInfo, в которую будет занесена информация о загруженном секретном ключе. Если указатель равен NULL, то параметр будет проигнорирован.

#### Возвращает:

Статус

SLBeginEnumKeysEx

Выполняет поиск сертификатов в таблице по критериям, описанным в структуре pEnumInfo. Незаполненные (обнулённые) поля при поиске не учитываются. Если заполнены несколько полей, то поиск ведётся по совпадению всех непустых полей (логическое «И»). Если ни одно поле не заполнено, то будет выдана информация по всем сертификатам.

Критерии совпадения полей:

1. Строковые поля совпадают, если строка в сертификате содержит подстроку, указанную в критерии поиска без учёта регистра.
6. Численные значения сравниваются на равенство.
7. Маски (битовые поля) совпадают, если операция логического «И» между критерием поиска и полем в сертификате даёт ненулевой результат.

Порядок использования функции:

6. Обнулить структуру. Указать в поле m\_cbSize структуры её размер в байтах.
8. Заполнить поля, по которым ведётся поиск.
9. Вызвать функцию.
10. Проверить код возврата. Если он успешный, то:
  - повторять вызов функции SLNextEnumKeys для получения информации о найденных ключах;
  - вызвать функцию SLEndEnumKeys для завершения поиска и освобождения ресурсов.

```
SECLIB_API(SLSTATUS)
SLBeginEnumKeysEx( HSECLIB hCtx, struct CSLKeyEnumInfoEx* pEnumInfo );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
pEnumInfo	Указатель на структуру с критериями поиска.

**Возвращает:**

Статус

**SLNextEnumKeyEx**

Возвращает информацию о найденном сертификате по его индексу. Для перебора всех найденных сертификатов необходимо, последовательно, начиная с нуля, увеличивая индекс, вызывать данную функцию до тех пор, пока она не вернёт код ошибки. Вызову данной функции должен предшествовать успешный вызов функции SLBeginEnumKeysEx.

Пример вывода на консоль списка найденных идентификаторов сертификатов и имён их владельцев:

```
unsigned          nIndex;
CSLKeyInfoEx     keyInfo;
SLSTATUS         Status;
for( nIndex = 0; ; ++nIndex )
{
    Status = SLNextEnumKeyEx( hCtx, nIndex, &keyInfo );
    if( SL_ERROR(Status) )
        break;
    printf( "%I64u %s\n", keyInfo.m_nKeyID, keyInfo.m_szName );
}
```

```
SECLIB_API(SLSTATUS)
SLNextEnumKeyEx( HSECLIB hCtx, unsigned nIndex, struct CSLKeyInfoEx* pKeyInfo );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
nIndex	Индекс сертификата.
pKeyInfo	Указатель на структуру типа CSLKeyInfoEx для возврата информации о найденном сертификате. Если указать значение NULL, то структура не будет заполняться.

**Возвращает:**

Статус

**SLEndEnumKeys**

Завершает операцию поиска ключей и освобождает занятые при поиске ресурсы.

```
SECLIB_API(SLSTATUS)
SLEndEnumKeys( HSECLIB hCtx );
```

**Параметры:**

---

---

Параметр	Назначение
hCtx	Дескриптор контекста.

**Возвращает:**

Статус

## Функции выработки и проверки цифровой подписи

### SLGetSignatureSize

Возвращает размер цифровой подписи в байтах.

```
SECLIB_API(unsigned)
SLGetSignatureSize(void);
```

**Параметры:**

нет

**Возвращает:**

размер цифровой подписи в байтах.

### SLBeginHashing

Подготавливает контекст к процедуре хэширования данных, необходимой для выработки/проверки цифровой подписи.

```
SECLIB_API(SLSTATUS)
SLBeginHashing( HSECLIB hCtx );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.

**Возвращает:**

Статус

### SLUpdateFileHash

Производит хэширование указанного участка файла. Хэшировать необходимо каждый участок файла, который подписывается.

```
SECLIB_API(SLSTATUS)
SLUpdateFileHash( HSECLIB hCtx, HANDLE hFile, unsigned long nOffset,
unsigned nDataSize );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
hFile	Дескриптор хэшируемого файла, открытого в режиме чтения.
nOffset	Смещение начала обрабатываемых данных в файле в байтах, начиная с нуля.
nDataSize	Размер хэшируемого участка данных в байтах

**Возвращает:**

Статус

**SLUpdateMemoryHash**

Производит хэширование указанного блока данных. Необходимо хэшировать каждый подписываемый участок данных.

SECLIB\_API(SLSTATUS)

SLUpdateMemoryHash( HSECLIB hCtx, const void\* pData, unsigned nDataSize );

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
pData	Указатель на хэшируемый блок данных
nDataSize	Размер хэшируемого блока данных в байтах

**Возвращает:**

Статус

**SLProduceSign**

Вырабатывает цифровую подпись ранее хэшированных данных. Размер цифровой подписи можно получить, вызвав функцию SLGetSignatureSize().

SECLIB\_API(SLSTATUS)

SLProduceSign( HSECLIB hCtx, void\* pSignature );

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
pSignature	Указатель на буфер, в который будет помещена выработанная цифровая подпись.

**Возвращает:**

Статус

**SLVerifySign**

Проверяет цифровую подпись ранее хэшированных данных.

SECLIB\_API(SLSTATUS)

SLVerifySign( HSECLIB hCtx, const void\* pSignature, struct CSLSignInfo\* pInfo );

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
pSignature	Указатель на цифровую подпись.
pInfo	Указатель на структуру типа CSLSignInfo для возврата информации о ключе, использовавшемся для проверки подписи. Структура не заполняется, если произошла ошибка поиска ключа или значение указателя равно NULL.

**Возвращает:**

Статус

## Функции шифрования данных

Для выполнения шифрования данных используется режим гаммирования ГОСТ 28147-89 с контролем целостности данных с помощью имитовставки по ГОСТ28147-89.

### SLGetEncryptionHdrSizeEx

Возвращает размер служебного заголовка шифрованных данных. Заголовок имеет переменную длину, зависящую от количества получателей сообщения.

SECLIB\_API(unsigned)

SLGetEncryptionHdrSizeEx( unsigned nRecipientCount );

#### Параметры:

Параметр	Назначение
nRecipientCount	Число получателей сообщения.

#### Возвращает:

Размер служебного заголовка в байтах

### SLSetDefaultKeyType

Устанавливает тип идентификатора ключа, используемого при зашифровании данных.

SECLIB\_API(SLSTATUS)

SLSetDefaultKeyType( HSECLIB hCtx, unsigned nKeyType );

#### Параметры:

Параметр	Назначение														
hCtx	Дескриптор контекста.														
nKeyType	Тип идентификатора ключа. Допускаются только значения, указанные в следующей таблице.														
<table border="1"> <thead> <tr> <th>Тип</th> <th>Описание</th> </tr> </thead> <tbody> <tr> <td>SL_KEYIDTYPE_OWNERID</td> <td>64-битовый идентификатор владельца ключа.</td> </tr> <tr> <td>SL_KEYIDTYPE_KEYID</td> <td>64-битовый идентификатор ключа.</td> </tr> <tr> <td>SL_KEYIDTYPE_OWNERNAME</td> <td>строка с именем владельца ключа.</td> </tr> <tr> <td>SL_KEYIDTYPE_OWNEREMAIL</td> <td>строка с электронным адресом владельца ключа.</td> </tr> <tr> <td>SL_KEYIDTYPE_EXTERNALID</td> <td>внешне заданный строковый идентификатор</td> </tr> <tr> <td>SL_KEYIDTYPE_EXTERNALNUMID</td> <td>внешне заданный числовой идентификатор</td> </tr> </tbody> </table>		Тип	Описание	SL_KEYIDTYPE_OWNERID	64-битовый идентификатор владельца ключа.	SL_KEYIDTYPE_KEYID	64-битовый идентификатор ключа.	SL_KEYIDTYPE_OWNERNAME	строка с именем владельца ключа.	SL_KEYIDTYPE_OWNEREMAIL	строка с электронным адресом владельца ключа.	SL_KEYIDTYPE_EXTERNALID	внешне заданный строковый идентификатор	SL_KEYIDTYPE_EXTERNALNUMID	внешне заданный числовой идентификатор
Тип	Описание														
SL_KEYIDTYPE_OWNERID	64-битовый идентификатор владельца ключа.														
SL_KEYIDTYPE_KEYID	64-битовый идентификатор ключа.														
SL_KEYIDTYPE_OWNERNAME	строка с именем владельца ключа.														
SL_KEYIDTYPE_OWNEREMAIL	строка с электронным адресом владельца ключа.														
SL_KEYIDTYPE_EXTERNALID	внешне заданный строковый идентификатор														
SL_KEYIDTYPE_EXTERNALNUMID	внешне заданный числовой идентификатор														

#### Возвращает:

Статус

## SECryptFileEx

Выполняет зашифрование участка файла в адрес нескольких (или одного) получателей. Список получателей задаётся массивом указателей на идентификаторы. Тип идентификаторов должен соответствовать типу установленному функцией `SLSetDefaultKeyIDType`. Вызывающая программа должна обеспечить передачу служебного заголовка и его размера приёмной стороне для использования при расшифровании. Размер данных после зашифрования не изменяется.

SECLIB\_API(SLSTATUS)

```
SECryptFileEx( HSECLIB hCtx, HANDLE hFileIn, HANDLE hFileOut,
unsigned long nInDataOffset, unsigned long nOutDataOffset, unsigned nDataSize, void*
ppRecipientList[], unsigned nRecipientCount, void* pvHeader );
```

### Параметры:

Параметр	Назначение
hCtx	Дескриптор контекста.
hFileIn	Дескриптор файла с открытыми данными.
hFileOut	Дескриптор файла для записи зашифрованных данных.
nInDataOffset	Смещение начала обрабатываемых данных в файле.
nOutDataOffset	Смещение, начиная с которого будут записаны зашифрованные данные.
nDataSize	Размер обрабатываемых данных.
ppRecipientList	Указатель на массив указателей на идентификаторы получателей.
nRecipientCount	Число получателей сообщения.
pvHeader	Указатель на буфер для возврата служебного заголовка. Размер буфера не должен быть меньше размера, возвращённого функцией <code>SLGetEncryptionHdrSizeEx()</code> для указанного числа получателей.

### Возвращает:

Статус

## SLDecryptFileEx

Выполняет расшифрование участка файла, зашифрованного в адрес одного или нескольких получателей. Вызывающая программа должна самостоятельно получить служебный заголовок и его размер. Размер данных после обработки не изменяется.

SECLIB\_API(SLSTATUS)

```
SLDecryptFileEx( HSECLIB hCtx, HANDLE hFileIn, HANDLE hFileOut,
unsigned long nInDataOffset, unsigned long nOutDataOffset, unsigned nDataSize,
const void* pHeader, unsigned nHeaderSize );
```

### Параметры:

Параметр	Назначение
hCtx	Дескриптор контекста.
hFileIn	Дескриптор файла с зашифрованными данными.
hFileOut	Дескриптор файла для записи расшифрованных данных.
nInDataOffset	Смещение начала обрабатываемых данных в файле.
nOutDataOffset	Смещение, начиная с которого будут записаны расшифрованные данные.
nDataSize	Размер обрабатываемых данных.
pvHeader	Указатель на служебный заголовок.

nHeaderSize	Размер служебного заголовка в байтах.
-------------	---------------------------------------

**Возвращает:**

Статус

**SLEncryptDataEx**

Выполняет зашифрование данных в памяти в адрес нескольких (или одного) получателей. Список получателей задаётся массивом указателей на идентификаторы. Тип идентификаторов должен соответствовать типу установленному функцией SLSetDefaultKeyIDType. Вызывающая программа должна обеспечить передачу служебного заголовка и его размера приёмной стороне для использования при расшифровании. Зашифрованные данные помещаются поверх исходных. Размер данных после зашифрования не изменяется.

SECLIB\_API(SLSTATUS)

```
SLEncryptFileEx( HSECLIB hCtx, void* pData, unsigned nDataSize, void* ppRecipientList[],
unsigned nRecipientCount, void* pvHeader );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
pData	Указатель на шифруемые данные. Результат зашифрования помещается поверх исходных данных.
nDataSize	Размер обрабатываемых данных.
ppRecipientList	Указатель на массив указателей на идентификаторы получателей.
nRecipientCount	Число получателей сообщения.
pvHeader	Указатель на буфер для возврата служебного заголовка. Размер буфера не должен быть меньше размера, возвращённого функцией SLGetEncryptionHdrSizeEx() для указанного числа получателей.

**Возвращает:**

Статус

**SLDecryptDataEx**

Выполняет расшифрование блока данных, зашифрованного в адрес одного или нескольких получателей. Вызывающая программа должна самостоятельно получить служебный заголовок и его размер. Размер данных после обработки не изменяется.

SECLIB\_API(SLSTATUS)

```
SLDecryptDataEx( HSECLIB hCtx, void* pData, unsigned nDataSize,
const void* pHeader, unsigned nHeaderSize );
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
pData	Указатель на блок зашифрованных данных, расшифрованные данные помещаются поверх исходных.
nDataSize	Размер обрабатываемых данных.
pvHeader	Указатель на служебный заголовок.
nHeaderSize	Размер служебного заголовка в байтах.

**Возвращает:**

Статус

**SL\_READ\_CALLBACK**

Прототип функции обратного вызова для выполнения чтения данных. Данная функция должна прочитать порцию данных размером не больше значения указанного в `pnDataSize`, затем поместить реальный размер прочитанных данных в эту же переменную. Если размер прочитанных данных меньше размера буфера, то считается, что это последний блок данных и последующих вызовов не будет.

```
typedef BOOL (__stdcall *SL_READ_CALLBACK)( void* pBuffer,
unsigned* pnDataSize, void* pCtx );
```

**Параметры:**

Параметр	Назначение
<code>pBuffer</code>	Указатель на буфер, в который необходимо прочитать данные.
<code>pnDataSize</code>	Указатель на переменную, которая содержит максимальный размер буфера. После выполнения чтения в неё необходимо занести реальный размер прочитанных данных.
<code>pCtx</code>	Значение пользовательского указателя, переданное при вызове функции зашифрования или расшифрования.

**Возвращает:**

TRUE – успешное завершение

FALSE – ошибка чтения.

**SL\_WRITE\_CALLBACK**

Прототип функции обратного вызова для выполнения записи данных. Эта функция должна записать блок данных указанного размера.

```
typedef BOOL (__stdcall *SL_WRITE_CALLBACK)( const void* pBuffer,
unsigned nDataSize, void* pCtx );
```

**Параметры:**

Параметр	Назначение
<code>pBuffer</code>	Указатель на буфер, который необходимо записать.
<code>nDataSize</code>	Размер записываемых данных.
<code>pCtx</code>	Значение пользовательского указателя, переданное при вызове функции зашифрования или расшифрования.

**Возвращает:**

TRUE – успешное завершение

FALSE – ошибка чтения.

**SLEncryptDataCB**

Выполняет зашифрование данных в памяти в адрес нескольких (или одного) получателей. Список получателей задаётся массивом указателей на идентификаторы. Тип идентификаторов должен соответствовать типу установленному функцией `SLSetDefaultKeyType`. Вызывающая программа должна обеспечить передачу служебного заголовка и его размера приёмной стороне для использования при расшифровании. Для чтения/записи данных используются функции обратного вызова. Размер данных после зашифрования не изменяется.

SECLIB\_API(SLSTATUS)

```
SLEncryptDataCB( HSECLIB hCtx, void* ppRecipientList[], unsigned nRecipientCount,
void* pvHeader, SL_READ_CALLBACK pfnReader, SL_WRITE_CALLBACK pfnWriter,
void* pCtx );;
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
ppRecipientList	Указатель на массив указателей на идентификаторы получателей.
nRecipientCount	Число получателей сообщения.
pvHeader	Указатель на буфер для возврата служебного заголовка. Размер буфера не должен быть меньше размера, возвращённого функцией <code>SLGetEncryptionHdrSizeEx()</code> для указанного числа получателей.
pfnReader	Указатель на функцию обратного вызова для чтения открытых данных.
pfnWriter	Указатель на функцию обратного вызова для записи зашифрованных данных.
pCtx	Пользовательский указатель, значение которого будет передаваться функциям обратного вызова.

**Возвращает:**

Статус

**SLDecryptDataCB**

Выполняет расшифрование блока данных, зашифрованного в адрес одного или нескольких получателей. Вызывающая программа должна самостоятельно получить служебный заголовок и его размер. Для чтения/записи данных используются функции обратного вызова. Размер данных после обработки не изменяется.

SECLIB\_API(SLSTATUS)

```
SLDecryptDataCB( HSECLIB hCtx, void* pvHeader, unsigned nHeaderSize,
SL_READ_CALLBACK pfnReader, SL_WRITE_CALLBACK pfnWriter,
void* pCtx );;
```

**Параметры:**

Параметр	Назначение
hCtx	Дескриптор контекста.
pvHeader	Указатель на служебный заголовок.
nHeaderSize	Размер служебного заголовка в байтах.
pfnReader	Указатель на функцию обратного вызова, для выполнения чтения зашифрованных данных.

---

---

pfnWriter	Указатель на функцию обратного вызова, для выполнения записи расшифрованных данных.
-----------	---

**Возвращает:**

Статус

## Структуры

### CSLSignInfo

Структура содержит информацию о цифровой подписи и владельце сертификата.

```
struct CSLSignInfo
{
    unsigned __int64          m_nKeyOwnerID;
    unsigned __int64          m_nKeyID;
    unsigned                m_nKeyRights;
    char                    m_szShortName[24];
    char                    m_szName[128];
    char                    m_szOrganization[128];
    char                    m_szDepartment[128];
    char                    m_szEMail[65];
    unsigned long           m_nSignDate;
    unsigned long           m_nSignTime;
};
```

#### Параметры:

Параметр	Назначение										
m_nKeyOwnerID	Идентификатор владельца ключа										
m_nKeyID	Идентификатор ключа										
m_nKeyRights	Права ключа. Могут быть значения, описанные константами вида SL_KEYRIGHT_XXX. Например, SL_KEYRIGHT_FIRSTSIGN. Более подробную информацию о ключе можно получить используя функции SLBeginEnumKeysEx/SLNextEnumKeys? указав в качестве критерия поиска идентификатор ключа.										
m_szShortName	Короткое имя владельца ключа.										
m_szName	Полное имя владельца ключа.										
m_szOrganization	Наименование организации										
m_szDepartment	Название отдела										
m_szEMail	Электронный адрес владельца.										
m_nSignDate	Дата выработки цифровой подписи. Формат представления даты: <table border="1" data-bbox="507 1350 1121 1498"> <thead> <tr> <th>Биты</th> <th>Назначение</th> </tr> </thead> <tbody> <tr> <td>0-7</td> <td>День (1...31)</td> </tr> <tr> <td>8-15</td> <td>Месяц (1...12)</td> </tr> <tr> <td>16-31</td> <td>Год (0 ... 65535)</td> </tr> </tbody> </table>	Биты	Назначение	0-7	День (1...31)	8-15	Месяц (1...12)	16-31	Год (0 ... 65535)		
Биты	Назначение										
0-7	День (1...31)										
8-15	Месяц (1...12)										
16-31	Год (0 ... 65535)										
m_nSignTime	Время выработки цифровой подписи. Формат представления времени: <table border="1" data-bbox="507 1621 1121 1805"> <thead> <tr> <th>Биты</th> <th>Назначение</th> </tr> </thead> <tbody> <tr> <td>0-7</td> <td>Секунды (0...59)</td> </tr> <tr> <td>8-15</td> <td>Минуты (0...59)</td> </tr> <tr> <td>16-23</td> <td>Часы (0...23)</td> </tr> <tr> <td>24-31</td> <td>Не используются</td> </tr> </tbody> </table>	Биты	Назначение	0-7	Секунды (0...59)	8-15	Минуты (0...59)	16-23	Часы (0...23)	24-31	Не используются
Биты	Назначение										
0-7	Секунды (0...59)										
8-15	Минуты (0...59)										
16-23	Часы (0...23)										
24-31	Не используются										

### CSLKeyInfoEx

Структура, описывающая информацию о ключе и его владельце.

```

struct CSLKeyInfoEx
{
    unsigned short          m_cbSize;
    unsigned __int64       m_nKeyOwnerID;
    unsigned __int64       m_nKeyID;
    unsigned               m_nKeyRights;
    char                   m_szShortName[24];
    char                   m_szName[128];
    char                   m_szOrganization[128];
    char                   m_szDepartment[128];
    char                   m_szEMail[65];
    char                   m_szExternalID[65];
    unsigned long          m_nGenDate;
    unsigned long          m_nRegDate;
    unsigned long          m_nStartDate;
    unsigned long          m_nEndDate;
    unsigned __int64       m_nExternalID;
    unsigned long          m_nKeyRightFlags;
    unsigned short         m_nKeyStatus;
};

```

**Параметры:**

Параметр	Назначение
m_cbSize	Размер структуры в байтах. Поле должно заполняться перед передачей структуры функциям библиотеки.
m_nKeyOwnerID	Идентификатор владельца ключа.
m_nKeyID	Идентификатор ключа
m_nKeyRights	Права ключа. Могут быть значения, описанные константами вида SL_KEYRIGHT_XXX. Например, SL_KEYRIGHT_FIRSTSIGN. Более подробную информацию о ключе можно получить используя функции SLBeginEnumKeysEx/SLNextEnumKeys? указав в качестве критерия поиска идентификатор ключа.
m_szShortName	Короткое имя владельца ключа.
m_szName	Полное имя владельца ключа.
m_szOrganization	Наименование организации
m_szDepartment	Название отдела
m_szEMail	Электронный адрес владельца.
m_szExternalID	Внешне задаваемый текстовый идентификатор ключа.
m_nGenDate	Дата и время генерации ключа в Unix-формате.
m_nRegDate	Дата и время регистрации ключа на ЦСК в Unix-формате.
m_nStartDate	Дата и время начала действия ключа в Unix-формате.
m_nEndDate	Дата и время окончания действия ключа в Unix-формате.
m_nExternalID	Внешне задаваемый числовой идентификатор ключа.
m_nKeyRightsFlag	Флаги, указывающие полномочия ключа. Установленный флаг указывает на наличие полномочия, сброшенный – на его отсутствие.
m_nKeyStatus	Текущее состояние ключа. Поле может содержать одно из значений, описанных константами SL_KEYSTATUS_XXX. Исключение составляет значение SL_KEYSTATUS_FLAG_LOCKED, которое используется как флаг.

**CSLKeyEnumInfoEx**

Структура, описывающая критерии поиска ключей..

```

struct CSLKeyEnumInfoEx
{
    unsigned short          m_cbSize;
    unsigned __int64       m_nKeyOwnerID;
    unsigned               m_nKeyRights;
};

```

```

char          m_szShortName[24];
char          m_szName[128];
char          m_szOrganization[128];
char          m_szDepartment[128];
char          m_szEMail[65];
char          m_szExternalID[65];
unsigned __int64 m_nExternalID;
unsigned long  m_nKeyRightFlags;
unsigned short m_nKeyStatus;
};

```

**Параметры:**

Параметр	Назначение
m_cbSize	Размер структуры в байтах. Поле должно заполняться перед передачей структуры функциям библиотеки.
m_nKeyOwnerID	Идентификатор владельца ключа.
m_nKeyRights	Права ключа. Могут быть значения, описанные константами вида SL_KEYRIGHT_XXX. Например, SL_KEYRIGHT_FIRSTSIGN. Более подробную информацию о ключе можно получить используя функции SLBeginEnumKeysEx/SLNextEnumKeys? указав в качестве критерия поиска идентификатор ключа.
m_szShortName	Короткое имя владельца ключа.
m_szName	Полное имя владельца ключа.
m_szOrganization	Наименование организации
m_szDepartment	Название отдела
m_szEMail	Электронный адрес владельца.
m_szExternalID	Внешне задаваемый текстовый идентификатор ключа.
m_nExternalID	Внешне задаваемый числовой идентификатор ключа.
m_nKeyRightsFlag	Флаги, указывающие полномочия ключа. Установленный флаг указывает на наличие полномочия, сброшенный – на его отсутствие.
m_nKeyStatus	Текущее состояние ключа. Поле может содержать одно из значений, описанных константами SL_KEYSTATUS_XXX. Исключение составляет значение SL_KEYSTATUS_FLAG_LOCKED, которое используется как флаг.

**Профили (типы) ключей**

Название флага	Наименование профиля (типа) ключа
SL_KEYRIGHT_OPER	Операционист
SL_KEYRIGHT_BUNG	Бухгалтер
SL_KEYRIGHT_BANK	Технолог ВПС (банк, филиалы, ББО и т.д.)
SL_KEYRIGHT_PERFORMER	Исполнитель
SL_KEYRIGHT_FIRSTSIGN	Юр. лицо 1-я подпись
SL_KEYRIGHT_SECONDSIGN	Юр. лицо 2-я подпись
SL_KEYRIGHT_PRIVATEEMPLOYER	Физическое лицо (частный предприниматель напр.)
SL_KEYRIGHT_ODB	Технолог АБС
SL_KEYRIGHT_JPKIAPPS	Технолог приложений

**Флаги, описывающие полномочия ключа.**

Название флага	Описание
SL_RIGHT_CERTIFICATE	Полномочие сертифицировать другие ключи (ключ сертификации).
SL_RIGHT_EMAIL	Полномочие защищать электронную почту.
SL_RIGHT_INTERNAL_DOCUMENTS	Полномочие защищать внутриорганизационные документы, т.е. такие документы, которые не выходят за пределы учреждения. Например, подпись операциониста в банке. Если документ отправляется наружу, то он должен быть переподписан с использованием другого ключа.
SL_RIGHT_EXTERNAL_DOCUMENTS	Полномочие защищать документы, которые могут передаваться другим организациям. Например, документ с подписью главного бухгалтера одного филиала банка может быть проверен в другом филиале.
SL_RIGHT_PAY_DOCUMENTS	Право подписывать платёжные документы.
SL_RIGHT_AUTENTIFICATION	Возможность использования сертификата для аутентификации владельца в системах управления доступом.
SL_RIGHT_BANK_CLIENT_KEYS	Сертификат предназначен для защиты документов в системе Клиент-Банк.
SL_RIGHT_BANK_KEYS	Сертификат предназначен для защиты документов платёжной системы банка.
SL_RIGHT_ENCODE_DATA	Сертификат может использоваться для шифрования данных.
SL_RIGHT_CLIENT_FIRST_SIGN	Право первой подписи юридического лица.
SL_RIGHT_CLIENT_SECOND_SIGN	Право второй подписи юридического лица.
SL_RIGHT_PEMPLOYER_SIGN	Право подписи физического лица.

**Значения, описывающие состояние ключа.**

Название флага	Описание
SL_KEYSTATUS_ACTUAL	Введен в действие.
SL_KEYSTATUS_CERTIFIED	Сертифицирован, но ещё не введен в действие.
SL_KEYSTATUS_NOT_ACTUAL	Выведен из действия.

SL_KEYSTATUS_COMPROMISED	Скомпрометирован (отозван).
SL_KEYSTATUS_NOT_CERTIFIED	Отказано в сертификации.
SL_KEYSTATUS_UPDATED	Введены в действие очередные ключи, но срок действия этого ключа еще не окончен.
SL_KEYSTATUS_FLAG_LOCKED	Временно заблокирован.

## Описание слова статуса.

Слово статуса возвращается большинством функций библиотеки для индикации результата её завершения.

### Формат слова статуса.

Биты	Назначение												
0-19	Код завершения, конкретное значение зависит от источника.												
20-29	Код источника. Определяет источник статуса. В данной версии возможны следующие источники: <table border="1" data-bbox="486 1037 1433 1373"> <thead> <tr> <th>Код источника</th> <th>Описание</th> </tr> </thead> <tbody> <tr> <td>SL_FACILITY_COMMON</td> <td>Общий для библиотеки источник</td> </tr> <tr> <td>SL_FACILITY_SECLIB</td> <td>Библиотека SecLib</td> </tr> <tr> <td>SL_FACILITY_KEYLIB</td> <td>Модуль управления ключами</td> </tr> <tr> <td>SL_FACILITY_C32CSP</td> <td>Модуль криптографии.</td> </tr> <tr> <td>SL_FACILITY_CERTLIB</td> <td>Модуль управления сертификатами.</td> </tr> </tbody> </table>	Код источника	Описание	SL_FACILITY_COMMON	Общий для библиотеки источник	SL_FACILITY_SECLIB	Библиотека SecLib	SL_FACILITY_KEYLIB	Модуль управления ключами	SL_FACILITY_C32CSP	Модуль криптографии.	SL_FACILITY_CERTLIB	Модуль управления сертификатами.
Код источника	Описание												
SL_FACILITY_COMMON	Общий для библиотеки источник												
SL_FACILITY_SECLIB	Библиотека SecLib												
SL_FACILITY_KEYLIB	Модуль управления ключами												
SL_FACILITY_C32CSP	Модуль криптографии.												
SL_FACILITY_CERTLIB	Модуль управления сертификатами.												

30-31	Код критичности статуса. Имеет следующие значения:	
	Значение	Описание
	SL_SEVERITY_SUCCESS	Успешное завершение.
	SL_SEVERITY_INFORMATIONAL	Успешное завершение, но с дополнительной информацией.
	SL_SEVERITY_WARNING	Предупреждение о возможной ошибочной ситуации. Интерпретация зависит от приложения, которое должно проанализировать код завершения и принять решение. Например, сообщение о проверке подписи на выведенном из действия ключе можно рассматривать двояко.
SL_SEVERITY_ERROR	Неуспешное завершение операции.	

### Макросы для проверки кода критичности.

Макрос	Назначение
SL_SUCCESS(Status)	Возвращает ненулевое значение для статуса со значением кода критичности не равным SL_SEVERITY_ERROR.
SL_INFORMATIONAL(Status)	Возвращает ненулевое значение для статуса со значением кода критичности равным SL_SEVERITY_INFORMATIONAL.
SL_WARNING(Status)	Возвращает ненулевое значение для статуса со значением кода критичности равным SL_SEVERITY_WARNING.
SL_ERROR(Status)	Возвращает ненулевое значение для статуса со значением кода критичности равным SL_SEVERITY_ERROR.

### Макросы для выделения полей статуса.

Макрос	Назначение
SL_SEVERITY(Status)	Возвращает только поле кода критичности, остальные поля обнулены.

---

---

SL_FACILITY(Status)	Возвращает только поле кода источника, остальные поля обнулены.
SL_CODE(Status)	Возвращает только поле кода завершения, остальные поля обнулены.