

ООО «ВАЛИДАТА»

УТВЕРЖДЕН
ВАМБ.00077-06-ЛУ

«ВАЛИДАТА КЛИЕНТ» ВЕРСИЯ 4

БИБЛИОТЕКА ПРИКЛАДНОГО ПРОГРАММНОГО ИНТЕРФЕЙСА
РАБОТЫ С СЕРТИФИКАТАМИ ДЛЯ ПЛАТФОРМЫ
MICROSOFT .NET FRAMEWORK

Руководство программиста

ВАМБ.00077-06 33 02

2020

Аннотация

Данный документ содержит описание библиотеки прикладного программного интерфейса работы с сертификатами для платформы Microsoft .Net Framework программного комплекса (ПК) ВАМБ.00077-06 «Валидата Клиент» версия 4» (далее по тексту — ПК «Валидата Клиент»), а также рекомендации по встраиванию и использованию данной библиотеки в прикладном программном обеспечении (ПО).

Документ предназначен для разработчиков прикладных программ (внешних по отношению к ПК «Валидата Клиент») как руководство по программированию с использованием библиотеки работы с сертификатами.

При встраивании библиотеки предполагается, что системный программист имеет знания о существующей архитектуре системы управления сертификатами (СУС), используемых рекомендациях и стандартах.

Содержание

1 ПРИКЛАДНОЙ ПРОГРАММНЫЙ ИНТЕРФЕЙС	4
1.1 Назначение	4
1.2 Характеристики	4
1.3 Использование	4
1.3.1 Состав	4
1.3.2 Подготовка среды разработки	4
1.3.3 Параметры конфигурации	5
1.4 Класс сертификата	5
1.5 Класс САС	8
1.6 Класс VcertException	9
1.7 Класс VcertObject	9
1.7.1 Методы инициализации	9
1.7.2 Метод деинициализации	11
1.7.3 Метод получения стека ошибок	11
1.7.4 Методы управления и вызова Мастеров	11
1.7.5 Справочники и профили пользователя	13
1.7.6 Визуализация объектов СУС	15
1.7.7 Экспорт и импорт объектов СУС	16
1.7.8 Разбор и получение информации об объектах СУС	18
1.7.9 Построение и проверка цепочек объектов СУС	20
1.7.10Вычисление хэш-значений	21
1.7.11Вычисление и проверка ЭП хэш-значений	23
1.7.12Поиск и перечисление объектов СУС	23
1.7.13Вычисление ЭП CMS-сообщений	25
1.7.14Проверка ЭП CMS-сообщений	28
1.7.15Зашифрование CMS-сообщений	34
1.7.16Расшифрование CMS-сообщений	36
1.7.17Преобразование совмещенных и отделенных ЭП	37
1.7.18Получение информации о CMS-сообщениях	39
1.7.19Простановка и проверка штампов времени	43
1.7.20Получение online-статуса сертификата	48
1.7.21Протокол безопасности транспортного уровня TLS 1.2	50
1.7.22Преобразование в формат и из формата Base64	52
1.7.23Выработка случайного числа заданной длины	53
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ	54
ПЕРЕЧЕНЬ РИСУНКОВ	55
ПЕРЕЧЕНЬ ТАБЛИЦ	56

1 ПРИКЛАДНОЙ ПРОГРАММНЫЙ ИНТЕРФЕЙС

1.1 Назначение

Библиотека прикладного программного интерфейса программного комплекса (ПК) ВАМБ.00077-06 «Валидата Клиент» версия 4» (далее — ПК «Валидата Клиент») для платформы Microsoft .NET Framework (далее по тексту — ППИ .NET) предоставляет программам, написанным с использованием Microsoft .NET Framework версий 3.5, 4.0, 4.5, 4.6, 4.7 доступ к набору констант, структур и функций библиотеки прикладного программного интерфейса для работы с сертификатами для С/С++ ПК «Валидата Клиент».

1.2 Характеристики

ППИ .NET предназначен для встраивания ПК «Валидата Клиент» в прикладные системы. Требования к аппаратно-программной среде, в которой функционирует ППИ .NET, приведены в документе ВАМБ.00077-06 30 01 «Валидата Клиент» версия 4. Формуляр».

При эксплуатации ППИ .Net необходимо обеспечивать те же требования к параметрам защиты от несанкционированного доступа, что и к прикладному программному обеспечению (ПО), использующему ПК «Валидата Клиент».

1.3 Использование

В данном документе дано краткое описание методов и классов с указанием соответствующих констант, структур и функций из библиотеки прикладного программного интерфейса для работы с сертификатами для С/С++.

Для полного описания методов и классов необходимо изучить описание соответствующих констант, структур и функций в документе ВАМБ.00077-06 33 01 «Валидата Клиент» версия 4. Руководство программиста».

Следует обратить внимание на то, что при выполнении потоковых операций с блоками памяти потоковая функция должна быть вызвана минимум два раза - один или несколько раз со значением параметра **IsLast = false** и один раз со значением **IsLast = true** (пусть и с входным блоком памяти нулевой длины).

1.3.1 Состав

В состав ППИ .NET входят следующие файлы:

– **vspia2.dll** – модуль динамической библиотеки.

Установка ППИ .NET выполняется при установке ПК «Валидата Клиент». Библиотека **vspia2.dll** устанавливается в глобальный кэш сборки (GAC) .NET.

1.3.2 Подготовка среды разработки

Разработка приложений для платформы Microsoft .NET Framework выполняется с помощью среды разработки Visual Studio. Перед началом сборки приложения с использованием ППИ .NET необходимо в проекте приложения добавить **vspia2.dll** в ссылки (References).

Классы, описанные в данном документе, находятся в пространстве имен (namespace) **Validata.PKI**.

В процессе разработки прикладного ПО может возникнуть необходимость детального анализа ошибочной ситуации, возникшей при вызове функций библиотеки. Для решения этой задачи библиотека поддерживает возможность включения отладочного протоколирования. Для включения отладочного протоколирования следует создать переменную окружения (Environment Variable) **pki1dbglvl**, установить ее значение в **0x80** и перезапустить процесс прикладного ПО.

Данные отладочного протокола записываются системной функцией **OutputDebugString()**. Для их просмотра можно воспользоваться отладчиком **WinDbg** или утилитой **DbgView** из состава пакета Windows SysInternals.

1.3.3 Параметры конфигурации

Библиотека прикладного программного интерфейса для работы с сертификатами для C/C++ позволяет выполнять настройки (изменять значения параметров конфигурации), предназначенные для адаптации к потребностям прикладного ПО, а также для увеличения производительности при использовании большого количества (1000000 и более) сертификатов.

Описание параметров конфигурации библиотеки приведено в документе ВАНБ.00077-06 33 01 «Валидата Клиент» версия 4. Руководство программиста».

1.4 Класс сертификата

Перечисление KeyUsages

Флаги возможной области использования открытого ключа:

– digitalSignature

соответствует флагу KEYUSAGE_DIGITAL_SIGNATURE.

– nonRepudiation

соответствует флагу KEYUSAGE_NON_REPUDIATION.

– keyEncipherment

соответствует флагу KEYUSAGE_KEY_ENCIPHERMENT.

– dataEncipherment

соответствует флагу KEYUSAGE_DATA_ENCIPHERMENT.

– keyAgreement

соответствует флагу KEYUSAGE_KEY_AGREEMENT.

– keyCertSign

соответствует флагу KEYUSAGE_KEY_CERT_SIGN.

– crlSign

соответствует флагу KEYUSAGE_CRL_SIGN.

– encipherOnly

соответствует флагу KEYUSAGE_ENCIPHER_ONLY.

– decipherOnly

соответствует флагу KEYUSAGE_DECIPHER_ONLY.

Класс KeyUsageExtension

Расширение область использования открытого ключа сертификата:

- свойство KeyUsages **Value**

Флаги возможной области использования открытого ключа.

- метод bool **Support** (KeyUsages **usage**)

Возвращает true, если область применения usage содержится в Value.

Класс AlternativeName (соответствует структуре *altname_t*)
Альтернативное имя издателя или владельца сертификата:

- свойство string **EmailAddress**

Строка с адресом электронной почты RFC 822.

- свойство string **DNS**

Строка с именем системы имен доменов DNS.

- свойство string **URI**

Строка с уникальным адресом ресурса URI.

- свойство string **IPAddress**

Строка с адресом IP протокола.

- свойство string **OrganizationName**

Строка с наименованием организации.

- свойство string **RegisteredAddress**

Строка с зарегистрированным адресом.

- свойство string **Surname**

Строка с Ф.И.О. владельца сертификата.

- свойство string **BusinessCategory**

Строка с должностью владельца сертификата.

- свойство string **TelephoneNumber**

Строка с номером телефона владельца сертификата.

- свойство string **Description**

Строка с описанием в свободной форме.

- свойство string **AccountNumber**

Строка с номером расчетного счета.

- свойство string **BankId**

Строка с банковским идентификационным кодом (БИК).

- свойство string **PhysicalDelivery**

Строка с почтовым адресом.

- свойство string **ExchangeAddress**

Строка с адресом Microsoft Exchange.

- свойство string **NotesAddress**

Строка с адресом Lotus Notes.

Класс Policy (соответствует структуре *policy_t*)
Регламент (политика) использования сертификата:

- свойство string **Oid**

Строка объектного идентификатора (OID), идентифицирующего регламент сертификата.

Класс ExtendedKeyUsage (соответствует структуре *extkeyusage_t*)
Расширенная область использования открытого ключа сертификата:

– свойство string **Oid**

Строка объектного идентификатора (OID), идентифицирующего использование открытого ключа сертификата.

Класс Extension (соответствует структуре *extension_t*)
Дополнение (расширение) X.509 сертификата:

– свойство string **Oid**

Строка объектного идентификатора (OID), идентифицирующего дополнение сертификата.

– свойство ExtensionType **Type**

ASN.1 тип дополнения.

– свойство bool **Critical**

Признак критичности дополнения.

– свойство byte[] **Value**

Дополнение в DER-кодировке.

Класс Certificate (соответствует структуре *certificate_t*)
Класс, описывающий сертификат:

– свойство string **Issuer**

X.500-имя издателя сертификата.

– свойство string **SerialNumber**

Серийный номер сертификата.

– свойство string **Subject**

X.500-имя владельца сертификата.

– свойство string **Algorithm**

Алгоритм открытого ключа сертификата.

– свойство DateTime **NotBefore**

Дата начала действия сертификата.

– свойство DateTime **NotAfter**

Дата окончания действия сертификата.

– свойство KeyUsageExtension **KeyUsage**

Разрешенные области использования открытого ключа.

– свойство DateTime **NotBeforePrivate**

Дата начала действия закрытого ключа сертификата.

– свойство DateTime **NotAfterPrivate**

Дата окончания действия закрытого ключа сертификата.

– свойство AlternativeName **IssuerAlternativeName**

Альтернативное имя издателя сертификата.

– свойство AlternativeName **SubjectAlternativeName**

Альтернативное имя владельца сертификата.

- свойство string **KeyIdentifier**

Строковый идентификатор закрытого ключа, соответствующего данному сертификату.

- свойство PolicyCollection **Policies**

Регламенты использования сертификата.

- свойство ExtendedKeyUsageCollection **ExtendedKeyUsage**

Расширенные области использования открытого ключа.

- свойство ExtensionCollection **Extensions**

Дополнения (расширения) сертификата.

- свойство bool **AlgorithmAsOid**

Если true, то поле Algorithm содержит объектный идентификатор (OID) алгоритма открытого ключа сертификата.

- свойство byte[] **EncodedCertificate**

Сертификат в DER-кодировке.

- свойство byte[] **CertificateHash**

Хэш-значение пары имени издателя и серийного номера сертификата.

1.5 Класс CAC

Класс RevCert (соответствует структуре revcert_t)

Аннулированный сертификат:

- свойство string **SerialNumber**

Серийный номер аннулированного сертификата.

- свойство DateTime **RevTime**

Время аннулирования сертификата.

- свойство int **Reason**

Причина аннулирования сертификата.

Класс CRL (соответствует структуре crl_t)

Класс, описывающий список аннулированных сертификатов (CAC):

- свойство string **Issuer**

X.500-имя издателя CAC.

- свойство DateTime **LastUpdate**

Время начала действия CAC.

- свойство DateTime **NextUpdate**

Время окончания действия CAC.

- свойство uint **Number**

Порядковый номер CAC.

- свойство RevCertCollection **RevCerts**

Аннулированные сертификаты.

- свойство byte[] **EncodedCrl**

CAC в DER-кодировке.

- свойство byte[] **CrlHash**

Данные, содержащиеся в дополнении «Идентификатор ключа владельца» сертификата издателя САС.

1.6 Класс VcertException

Свойство uint **Error**

Код ошибки.

Примечание - Коды ошибок, а также детальное описание и причина возникновения ошибок описаны в разделе «Описание ошибочных ситуаций» документа ВАМБ.00077-06 33 01 «Валидата Клиент» версия 4. Руководство программиста».

Свойство string **Message**

Описание ошибки.

1.7 Класс VcertObject

Класс **VcertObject** является классом-оберткой контекста локальной библиотеки. При вызове методов класса **VcertObject** в случае возникновения ошибки интерфейса генерируется исключение **VcertException**.

Примечание - В процессе своей работы приложению разрешается инициализировать и деинициализировать несколько различных полноценных контекстов библиотеки, в том числе в целях их одновременного (параллельного) использования.

1.7.1 Методы инициализации

Перечисление InitializeFlags

Флаги инициализации контекста библиотеки:

– DisableCrlUpdate

соответствует флагу FLAG_INIT_DISABLECRLUPDATE.

– ExpiringObjectUI

соответствует флагу FLAG_INIT_EXPIRINGOBJECTUI.

– DisableLdapUsage

соответствует флагу FLAG_INIT_DISABLELDAPUSAGE.

– DisableCacheSave

соответствует флагу FLAG_INIT_DISABLECACHESAVE.

– CertStoreProfile

соответствует флагу FLAG_INIT_CERTSTOREPROFILE.

– ForceCertCaching

соответствует флагу FLAG_INIT_FORCECERTCACHING.

– AllowAiaCdpUsage

соответствует флагу FLAG_INIT_ALLOWAIACDPUSAGE.

- UseSilentKeyLoad

соответствует флагу FLAG_INIT_USESILENTKEYLOAD.

- UseVerifyContext

соответствует флагу FLAG_INIT_USEVERIFYCONTEXT.

Перед использованием любого метода класса VcertObject (за исключением функций инициализации) его необходимо инициализировать с помощью одной из функций инициализации.

bool **InitMinimal()** - *(соответствует функции VCERT_InitMinimal)*

Метод инициализации минимального контекста библиотеки

Возвращаемые значения:

- **true** в случае успеха.

bool **Initialize**(string profile, InitializeFlags flags) - *(соответствует функции VCERT_Initialize)*

Метод инициализации контекста библиотеки

Аргументы:

– **profile** строка с именем профиля из настроек конфигурационного файла **pki1.conf** или из настроек профилей пользователя ПК «Справочник сертификатов» при указании флага **InitializeFlags.CertStoreProfile**. Поддерживаются следующие дополнительные возможности:

- задание значения «МУ» позволяет использовать настройки профилей пользователя ПК «Справочник сертификатов» и пользовательский интерфейс выбора профиля без задания флага **InitializeFlags.CertStoreProfile**;

- задание флага инициализации **InitializeFlags.CertStoreProfile** позволяет указать требуемый профиль пользователя ПК «Справочник сертификатов» по имени;

- **flags** флаги инициализации (см. описание в п. 1.7.1).

Возвращаемые значения:

- **true** в случае успеха.

bool **Initialize**(string pse, string localstore, string ldap, string pincode, InitializeFlags flags) - *(соответствует функции VCERT_InitializeEx)*

Расширенный метод инициализации контекста библиотеки

Аргументы:

- **pse** Путь (URI) к персональному справочнику пользователя (ПСП) *(соответствует полю **pse** структуры **local_param_t**).*

- **localstore** Путь (URI) к локальному справочнику пользователя (ЛСП) *(соответствует полю **localstore** структуры **local_param_t**).*

- **ldap** Путь (URI) к сетевому справочнику сертификатов (ССС) *(соответствует полю **ldap** структуры **local_param_t**).*

- **pincode** ПИН-код ключевого носителя типа смарт-карта *(соответствует*

полю **pincode** структуры **local_param_t**).

- **flags** флаги инициализации (см. описание в п. 1.7.1).

Возвращаемые значения:

- **true** в случае успеха.

1.7.2 Метод деинициализации

void **Uninitialize()** - (соответствует функции *VCERT_Uninitialize*)

Метод деинициализации (освобождения) контекста библиотеки

Данный метод выполняет деинициализацию (освобождение) контекста библиотеки и связанных с ним ресурсов, выделенных в методах *InitMinimal()*, *Initialize()*, а также производит выгрузку загруженных для этого контекста закрытых ключей. Данный метод вызывается в деструкторе класса **VcertObject**.

1.7.3 Метод получения стека ошибок

string **GetErrorLineEx()** - (соответствует функции *VCERT_GetErrorLineEx*)

Метод получения стека внутренних ошибок библиотеки

Возвращаемые значения:

- Стек внутренних ошибок библиотеки.

1.7.4 Методы управления и вызова Мастеров

Перечисление **VcertCmd**

Команда, которую необходимо выполнить:

- **UpdateCRLs**

соответствует команде *VCERT_CMD_UPDATECRLS*.

- **UpdateCRLsCrit**

соответствует команде *VCERT_CMD_UPDATECRLSCRIT*.

- **SignWizard**

соответствует команде *VCERT_CMD_SIGN_WIZARD*.

- **VerifyWizard**

соответствует команде *VCERT_CMD_VERIFY_WIZARD*.

- **XmlReqWizard**

соответствует команде *VCERT_CMD_XMLREQ_WIZARD*.

- **ProfileWizard**

соответствует команде *VCERT_CMD_PROFILE_WIZARD*.

- **PrivateKeyWizard**

соответствует команде *VCERT_CMD_PRIVKEY_WIZARD*.

- **CarrierSetPin**

соответствует команде *VCERT_CMD_CARRIER_SETPIN*.

- **CarrierFormat**

соответствует команде *VCERT_CMD_CARRIER_FORMAT*.

Перечисление CtrlFlags

Флаги выполнения команды:

- XmlReqForEncryption

соответствует флагу FLAG_CTRL_XMLREQ_FOR_ENCRYPTION.

- XmlReqGOST_R_34_10_12_256

соответствует флагу FLAG_CTRL_XMLREQ_GOST_R_34_10_12_256.

- XmlReqGOST_R_34_10_12_512

соответствует флагу FLAG_CTRL_XMLREQ_GOST_R_34_10_12_512.

- XmlReqCarrierGeneration

соответствует флагу FLAG_CTRL_XMLREQ_CARRIER_GENERATION.

- ProfWizProhibitModifies

соответствует флагу FLAG_CTRL_PROFWIZ_PROHIBIT_MODIFIES.

void **ControlEx**(IntPtr hWnd, VcertCmd cmd, byte[] idat, out byte[] odat, CtrlFlags flag) - (*соответствует функции VCERT_ControlEx*)

Метод управления контекстом и библиотекой, вызов Мастеров

Аргументы:

- **hWnd** идентификатор (опциональный) родительского окна;
- **cmd** код команды, которую необходимо выполнить;
- **idat** блок памяти с входными данными (зависит от выполняемой команды);
- **odat** блок памяти с выходными данными (зависит от выполняемой команды);
- **flag** флаги выполнения (зависят от выполняемой команды).

void **UpdateCRLs**() - (*соответствует функции VCERT_UpdateCRLs*)

Метод стандартного обновления всех САС, найденных в ЛСП, из точек CDP
 Данная функция выполняет вызов функции управления контекстом **VCERT_ControlEx()** с кодом команды **VCERT_CMD_UPDATECRLS**.

Перечисление XmlRequestFlags

Флаги формирования XML-запроса:

- DoNotGenerate

соответствует флагу FLAG_XML_REQUEST_DO_NOT_GENERATE.

- GOST_R_34_10_12_256

соответствует флагу FLAG_XML_REQUEST_GOST_R_34_10_12_256.

- GOST_R_34_10_12_512

соответствует флагу FLAG_XML_REQUEST_GOST_R_34_10_12_512.

- CarrierGeneration

соответствует флагу FLAG_XML_REQUEST_CARRIER_GENERATION.

void **CreateXmlRequest**(XmlRequestFlags flag, Certificate certTmpl, out byte[] xmlRequest) - (*соответствует функции VCERT_CreateXmlRequest*)

Метод создания парных ключей и XML запроса по шаблону сертификата**Аргументы:**

- **flag** флаг формирования XML-запроса:
- **certTpl** класс сертификата - шаблон, на основании которого формируется XML-запрос.
- **xmlRequest** блок памяти с сформированным XML-запросом.

1.7.5 Справочники и профили пользователя**Класс memblockCollection**Коллекция блоков памяти

- метод **int Add**(byte[] block) - добавить блок памяти в коллекцию, возвращает индекс в коллекции;
- свойство **uint Cnt** - количество блоков памяти.

void CreateCertificateStore(memblockCollection certs, memblockCollection crls, string pstore, string localstore, uint flag) - (соответствует функции *VCERT_CreateCertificateStore*)

Метод формирования ПСП и ЛСП из отдельных сертификатов и САС**Аргументы:**

- **certs** коллекция блоков памяти, содержащих сертификаты в DER-кодировке или PEM-формате. Первый элемент данного массива должен содержать рабочий сертификат пользователя с действующим ключом ЭП. В формируемые ПСП и ЛСП будут включены только те сертификаты, которые на момент формирования действительны по времени;
- **crls** коллекция блоков памяти, содержащих САС в DER-кодировке или PEM-формате. В формируемые ПСП и ЛСП будут включены только те САС, которые на момент формирования действительны по времени;
- **pstore** путь (URI) к создаваемому ПСП;
- **localstore** путь (URI) к создаваемому ЛСП;
- **flag** маска флагов (зарезервировано, должно быть равно **0**).

Перечисление ProfileFlagsФлаги методов работы с профилями:

- **CreateRegistryProfileOver**
соответствует флагу *FLAG_CREATE_REGISTRY_PROFILE_OVER*.

bool CreateRegistryProfile(string profile, string storepath, ProfileFlags flags) - (соответствует функции *VCERT_CreateRegistryProfile*)

Метод создания или обновления профиля по пути к файлам ПСП и ЛСП**Аргументы:**

- **profile** имя создаваемого или обновляемого профиля пользователя, например **По умолчанию**;
- **storepath** путь к файлам ПСП **local.pse** и ЛСП **local.gdbm**. Данную функ-

цию можно использовать только в тех случаях, когда ПСП не расположен в системном хранилище ОС Microsoft Windows, а ЛСП расположен в базе данных GDBM;

- **flag** флаг добавления или обновления профиля.

Возвращаемые значения:

- **true** в случае успеха.

bool **CreateRegistryProfileEx**(string profile, string pstore, string localstore, string ldapstore, ProfileFlags flags) - (соответствует функции VCERT_
CreateRegistryProfileEx)

Метод создания или обновления профиля по путям (URI) к ПСП, ЛСП, ССС

Аргументы:

- **profile** имя создаваемого или обновляемого профиля пользователя, например **По умолчанию**;

- **pstore** путь (URI) к ПСП профиля пользователя;
- **localstore** путь (URI) к ЛСП профиля пользователя;
- **ldapstore** путь (URI) (опциональный) к ССС профиля пользователя;
- **flag** флаг добавления или обновления профиля.

Возвращаемые значения:

- **true** в случае успеха.

string **QueryRegistryProfile**(uint index) - (соответствует функции VCERT_
QueryRegistryProfile)

Метод получения информации о профиле по индексу

Аргументы:

- **index** индекс профиля.

Возвращаемые значения:

- имя профиля.

void **QueryRegistryProfileEx**(out string profile, out string pstore, out string localstore, out string ldapstore, uint index) - (соответствует функции VCERT_
QueryRegistryProfileEx)

Расширенный метод получения информации о профиле по индексу

Аргументы:

- **profile** имя профиля;
- **pstore** путь (URI) к ПСП;
- **localstore** путь (URI) к ЛСП;
- **ldapstore** путь (URI) к ССС;
- **index** индекс профиля.

1.7.6 Визуализация объектов СУС

void **ShowCertificate**(byte[] cert) - (*соответствует функции VCERT_ShowCertificate*)

Метод визуализации сертификата

Аргументы:

- **cert** блок памяти с сертификатом в DER-кодировке или PEM-формате.

void **ShowCertificateEx**(IntPtr hWnd, byte[] cert) - (*соответствует функции VCERT_ShowCertificateEx*)

Расширенный метод визуализации сертификата

Аргументы:

- **hWnd** идентификатор родительского окна;
- **cert** блок памяти с сертификатом в DER-кодировке или PEM-формате.

void **ShowCrl**(byte[] crl) - (*соответствует функции VCERT_ShowCrl*)

Метод визуализации САС

Аргументы:

- **crl** блок памяти с САС в DER-кодировке или PEM-формате.

void **ShowCrlEx**(IntPtr hWnd, byte[] crl) - (*соответствует функции VCERT_ShowCrlEx*)

Расширенный метод визуализации САС

Аргументы:

- **hWnd** идентификатор родительского окна;
- **crl** блок памяти с САС в DER-кодировке или PEM-формате.

void **ShowRequestEx**(IntPtr hWnd, byte[] req) - (*соответствует функции VCERT_ShowRequestEx*)

Расширенный метод визуализации PKCS#10 запроса

Аргументы:

- **hWnd** идентификатор родительского окна;
- **req** блок памяти с PKCS#10 запросом в DER-кодировке или PEM-формате.

void **ShowRevocationEx**(IntPtr hWnd, byte[] rev) - (*соответствует функции VCERT_ShowRevocationEx*)

Расширенный метод визуализации запроса на аннулирование сертификата

Аргументы:

- **hWnd** идентификатор родительского окна;
- **rev** блок памяти с запросом на аннулирование сертификата в DER-кодировке.

1.7.7 Экспорт и импорт объектов СУС

Перечисление **ExportParameters.ExportObjectType**

Тип экспортируемого объекта:

- **NewRequest** - выполнить формирование запроса PKCS#10 для проведения плановой смены рабочего сертификата пользователя;
- **RevocationRequest** - выполнить формирование запроса на аннулирование рабочего сертификата при компрометации закрытого ключа пользователя.

Класс **ExportParameters** - (соответствует структуре *export_param_t*)

Параметры формирования запроса PKCS#10 или запроса на аннулирование:

- Конструктор **ExportParameters**(ExportParameters.ExportObjectType type);
- свойство bool **ShowUI** - автоматически визуализировать сформированный запрос PKCS#10 или запрос на аннулирование;
- свойство bool **UseDER** - экспортировать запрос PKCS#10 или запрос на аннулирование в DER-кодировке, без оберты в формате CMS;
- свойство bool **GenKey2012_256** - выполнить генерацию закрытого ключа для сертификата в квалифицированном формате по ГОСТ Р 34.10-2012 (256 бит);
- свойство bool **GenKey2012_512** - выполнить генерацию закрытого ключа для сертификата в квалифицированном формате по ГОСТ Р 34.10-2012 (512 бит);
- свойство bool **CarrierGeneration** - при генерации неизвлекаемого закрытого ключа на функциональном ключевом носителе (ФКН) vdToken формировать этот ключ внутри ФКН с использованием внутреннего ДСЧ последнего.

byte[] **ExportMemory**(ExportParameters Param) - (соответствует функции VCERT_ExportMem)

Метод формирования блока памяти с запросом PKCS#10/на аннулирование

Аргументы:

- **Param** параметры формирования запроса PKCS#10/на аннулирование.

Возвращаемые значения:

- блок данных с сформированным запросом.

void **ExportFile**(ExportParameters Param, string outfile) - (соответствует функции VCERT_ExportFile)

Метод формирования файла с запросом PKCS#10/на аннулирование

Аргументы:

- **Param** параметры формирования запроса PKCS#10/на аннулирование;
- **outfile** путь к файлу для записи сформированного запроса.

Перечисление **ExportToStoreFlags**

Флаги экспорта сертификатов и САС в системное хранилище ОС:

- LocalMachineStore

соответствует флагу FLAG_EXPORT_SYSTEM_STORE_MACHINE.

- IgnoreErrors

соответствует флагу FLAG_EXPORT_SYSTEM_STORE_IGNORE.

void **ExportToSystemStore**(ExportToStoreFlags flag) - (соответствует функции VCERT_ExportToSystemStore)

Метод экспорта сертификатов и САС из ПСП и ЛСП в системное хранилище ОС

Аргументы:

- **flag** флаги экспорта сертификатов и САС.

Перечисление ImportParameters.ImportObjectType

Тип импортируемого объекта:

- **Certificate** - импортировать сертификат в DER-кодировке или PEM-формате;

- **Crl** - импортировать САС в DER-кодировке или PEM-формате;

- **MyCertificate** - импортировать сертификат в DER-кодировке или PEM-формате и установить его в качестве рабочего сертификата пользователя;

- **Update** - импортировать сертификат или САС без построения и проверки цепочки сертификации.

Класс **ImportParameters** - (соответствует структуре import_param_t)

Параметры добавления объектов СУС в ПСП, ЛСП или ССС:

- Конструктор **ImportParameters**(ImportParameters.ImportObjectType type);

- свойство bool **ShowUI** - автоматически визуализировать импортируемые в ПСП, ЛСП или ССС объекты СУС;

- свойство bool **DoNotVerifyChain** - импортировать сертификат или САС без построения и проверки цепочки сертификации;

- свойство bool **ForceRemoteStore** - принудительно импортировать сертификат или САС в ССС.

void **ImportMemory**(ImportParameters Param, byte[] Data) - (соответствует функции VCERT_ImportMem)

Метод добавления объектов СУС в ПСП, ЛСП или ССС из блока данных

Аргументы:

- **Param** параметры добавления объектов СУС;

- **Data** блок данных с добавляемыми объектами в DER-кодировке или PEM-формате.

void **ImportFile**(ImportParameters Param, string Data) - (соответствует функции VCERT_ImportFile)

Метод добавления объектов СУС в ПСП, ЛСП или ССС из файла

Аргументы:

- **Param** параметры добавления объектов СУС;
- **Data** путь к файлу с добавляемыми объектами в DER-кодировке или PEM-формате.

1.7.8 Разбор и получение информации об объектах СУС

Методы разбора и получения информации предназначены для преобразования (разбора) закодированных объектов СУС из DER-кодировки или PEM-формата в соответствующие данным объектам структуры.

Методы также можно использовать для разбора дополнений (расширений) сертификатов из DER-кодировки. Для разбора дополнения (расширения) его необходимо найти в массиве дополнений (расширений) сертификата по объектному идентификатору (OID), и передать блок памяти с закодированными данными дополнения в соответствующий метод разбора:

- альтернативное имя владельца (OID 2.5.29.17) - метод **ParseAltnameEx()**;
- альтернативное имя издателя (OID 2.5.29.18) - метод **ParseAltnameEx()**;
- базовые ограничения сертификата (OID 2.5.29.19) - метод

ParseBasicConstraintsEx();

- идентификатор ключа издателя (OID 2.5.29.35) - метод

ParseKeyIdentifierEx().

Класс **OtherName** (соответствует структуре *othername_t*)

Другое имя альтернативного имени:

- свойство string **Oid**

Строка с объектным идентификатором (OID), идентифицирующим другое имя.

- свойство string **Name**

Строка с текстовыми данными другого имени.

Класс **AltNameEx** (соответствует структуре *altname_ex_t*)

Другие имена альтернативного имени:

- свойство OtherNameCollection **OtherNames**

Поле с массивом других имен альтернативного имени.

Класс **BasicConstraintsEx** (соответствует структуре *basic_constraints_ex_t*)

Базовые ограничения сертификата:

- свойство uint **CA**

Признак сертификата Центра сертификации (ЦС). Если значение **!= 0**, то сертификат является сертификатом ЦС.

- свойство int **PathLen**

Максимальная длина цепочки сертификации. Значение **< 0** означает отсут-

ствие ограничения на длину цепочки.

Certificate **ParseCertificate**(byte[] Encoded) - (соответствует функции *VCERT_ParseCert*)

Метод разбора и получения информации о сертификате

Аргументы:

- **Encoded** блок памяти с сертификатом в DER-кодировке или PEM-формате.

Возвращаемые значения:

- класс сертификата.

CRL **ParseCrl**(byte[] Encoded) - (соответствует функции *VCERT_ParseCrl*)

Метод разбора и получения информации о САС

Аргументы:

- **Encoded** блок памяти с САС в DER-кодировке или PEM-формате.

Возвращаемые значения:

- класс САС.

AltNameEx **ParseAltnameEx**(byte[] Encoded) - (соответствует функции *VCERT_ParseAltnameEx*)

Метод разбора и получения информации об альтернативном имени

Аргументы:

- **Encoded** блок памяти с данными дополнения альтернативного имени в DER-кодировке.

Возвращаемые значения:

- класс альтернативного имени.

BasicConstraintsEx **ParseBasicConstraintsEx**(byte[] Encoded) - (соответствует функции *VCERT_ParseBasicConstraintsEx*)

Метод разбора и получения информации о базовых ограничениях сертификата

Аргументы:

- **Encoded** блок памяти с данными дополнения базовых ограничений сертификата в DER-кодировке.

Возвращаемые значения:

- класс базовых ограничений.

byte[] **ParseKeyIdentifierEx**(byte[] Ikid) - (соответствует функции *VCERT_ParseKeyIdentifierEx*)

Метод разбора и получения информации об идентификаторе ключа издателя

Аргументы:

- **Ikid** блок памяти с данными дополнения идентификатора ключа издателя в DER-кодировке.

Возвращаемые значения:

- блок памяти с разобранным идентификатором ключа издателя.

1.7.9 Построение и проверка цепочек объектов СУС

Перечисление **VerifyPolicyParameters.VerifyPolicyFlags**

Флаги построения и проверки цепочки сертификата или САС:

- NoTimeCheck

соответствует флагу FLAG_POLICY_DONOTCHECKTIMES.

- DoCrlVerification

соответствует флагу FLAG_POLICY_DOCRLVERIFICATION.

- DoNotCheckKeyTime

соответствует флагу FLAG_POLICY_DONOTCHECKKEYTIME.

Класс **VerifyPolicyParameters**(соответствует структуре *verify_policy_param_t*)

Параметры построения и проверки цепочки сертификата или САС:

- свойство VerifyPolicyFlags **Flags**

флаги построения и проверки цепочки сертификата или САС.

- свойство int **CheckTime**

момент времени, на который строится и проверяется цепочка сертификата.

- KeyUsages **KeyUsage**

разрешенные области использования открытого ключа.

- метод void **AddExtendedKeyUsage**(string eku)

Добавление OID расширенной области использования открытого ключа для проверки.

- метод void **AddPolicy**(string policy)

Добавление OID регламента использования сертификата для проверки.

Certificate **VerifyCert**(CmsVerifyParameters Param, byte[] Cert) - (соответствует функции *VCERT_VerifyCert*)

Метод построения и проверки цепочки сертификата

Аргументы:

- **Param** параметры проверки сертификата (см. п. 1.7.14);

– **Cert** блок памяти с проверяемым сертификатом в DER-кодировке или PEM-формате.

Возвращаемые значения:

- класс сертификата.

bool **VerifyCertificatePolicy**(VerifyPolicyParameters Param, byte[] Cert) - (соответствует функции *VCERT_VerifyCertificatePolicy*)

Расширенный метод построения и проверки цепочки сертификата

Аргументы:

- **Param** параметры построения и проверки цепочки сертификата;

– **Cert** блок памяти с проверяемым сертификатом в DER-кодировке или PEM-формате.

Возвращаемые значения:

– **true** в случае успеха.

bool **VerifyCrlPolicy**(VerifyPolicyParameters Param, byte[] Crl) - (соответствует функции *VCERT_VerifyCrlPolicy*)

Расширенный метод построения и проверки цепочки САС

Аргументы:

– **Param** параметры построения и проверки цепочки САС (необходимо установить флаг **DoCrlVerification**);

– **Crl** блок памяти с проверяемым САС в DER-кодировке или PEM-формате.

Возвращаемые значения:

– **true** в случае успеха.

1.7.10 Вычисление хэш-значений

byte[] **BlkHashMem**(string Algorithm, byte[] Data) - (соответствует функции *VCERT_BlkHashMem*)

Метод блочного вычисления хэш-значения блока памяти

Аргументы:

– **Algorithm** строка объектного идентификатора (OID) алгоритма хэширования (в случае, если заданное значение не равно ни одному из перечисленных ниже, хэширование будет выполняться по ГОСТ Р 34.11-94):

- "1.2.643.2.2.9" - хэширование по ГОСТ Р 34.11-94;
- "1.2.643.7.1.1.2.2" - хэширование по ГОСТ Р 34.11-2012 (256 бит);
- "1.2.643.7.1.1.2.3" - хэширование по ГОСТ Р 34.11-2012 (512 бит);

– **Data** блок памяти с исходными данными (блок памяти может быть нулевой длины).

Возвращаемые значения:

– вычисленное хэш-значение.

byte[] **BlkHashFile**(string Algorithm, string Data) - (соответствует функции *VCERT_BlkHashFile*)

Метод блочного вычисления хэш-значения файла

Аргументы:

– **Algorithm** строка объектного идентификатора (OID) алгоритма хэширования (в случае, если заданное значение не равно ни одному из перечисленных ниже, хэширование будет выполняться по ГОСТ Р 34.11-94):

- "1.2.643.2.2.9" - хэширование по ГОСТ Р 34.11-94;
- "1.2.643.7.1.1.2.2" - хэширование по ГОСТ Р 34.11-2012 (256 бит);

- "1.2.643.7.1.1.2.3" - хэширование по ГОСТ Р 34.11-2012 (512 бит);

– **Data** имя файла с исходными данными (файл не может быть нулевой длины).

Возвращаемые значения:

- вычисленное хэш-значение.

Класс **StreamHashCtx**

Контекст выполнения потоковой операции:

- Конструктор **StreamHashCtx**(string hashalg);
- Свойство string **algorithm** строка объектного идентификатора (OID) алгоритма хэширования:

- "1.2.643.2.2.9" - хэширование по ГОСТ Р 34.11-94;
- "1.2.643.7.1.1.2.2" - хэширование по ГОСТ Р 34.11-2012 (256 бит);
- "1.2.643.7.1.1.2.3" - хэширование по ГОСТ Р 34.11-2012 (512 бит).

byte[] **StreamHashMem**(StreamHashCtx hashContext, byte[] Data, bool IsLast) - (*соответствует функциям VCERT_StrHashInitMem, VCERT_StrHashUpdateMem, VCERT_StrHashFinalMem*)

Метод потокового вычисления хэш-значения блоков памяти

Аргументы:

- **hashContext** контекст выполнения потоковой операции;
- **Data** блок памяти с исходными данными (блок памяти может быть нулевой длины);
- **IsLast** признак завершения потокового вычисления хэш-значения.

Возвращаемые значения:

- вычисленное хэш-значение, возвращается при завершении потокового вычисления хэш-значения.

byte[] **StreamHashFile**(string Algorithm, string Data) - (*соответствует функции VCERT_StrHashFile*)

Метод потокового вычисления хэш-значения файла

Аргументы:

- **Algorithm** строка объектного идентификатора (OID) алгоритма хэширования:
- "1.2.643.2.2.9" - хэширование по ГОСТ Р 34.11-94;
- "1.2.643.7.1.1.2.2" - хэширование по ГОСТ Р 34.11-2012 (256 бит);
- "1.2.643.7.1.1.2.3" - хэширование по ГОСТ Р 34.11-2012 (512 бит);
- **Data** имя файла с исходными данными (файл не может быть нулевой длины).

Возвращаемые значения:

- вычисленное хэш-значение.

1.7.11 Вычисление и проверка ЭП хэш-значений

byte[] **SignHash**(byte[] Hash) - (соответствует функции VCERT_SignHashMem)

Метод вычисления ЭП хэш-значения

Аргументы:

- **hash** блок памяти с хэш-значением;

Возвращаемые значения:

- вычисленное ЭП хэш-значения.

bool **VerifyHash**(byte[] Signer, byte[] Hash, byte[] Sign) - (соответствует функции VCERT_VerifyHashMem)

Функция проверки ЭП хэш-значения

Аргументы:

– **Signer** блок памяти с сертификатом, на котором необходимо проверять ЭП (в DER-кодировке или PEM-формате);

- **Hash** блок памяти с хэш-значением;

- **Sign** блок памяти с проверяемой ЭП хэш-значения.

Возвращаемые значения:

- **true** в случае успеха.

1.7.12 Поиск и перечисление объектов СУС

Перечисление **FindParameters.FindFlags**

Флаги поиска сертификатов:

- FindMy

соответствует флагу FLAG_FIND_CERTWITHPRIVATE.

- FindRemote

соответствует флагу FLAG_FIND_USEREMOTESEARCH.

- FindSelectUi

соответствует флагу FLAG_FIND_SHOWUISELECTOR.

- NoTimeCheck

соответствует флагу FLAG_FIND_DONOTCHECKTIMES.

- NoVerify

соответствует флагу FLAG_FIND_DONOTVERIFYCHAIN.

- Ldap2Local

соответствует флагу FLAG_FIND_ADDREMOTETOLOCAL.

- NoKeyTimeCheck

соответствует флагу FLAG_FIND_DONOTCHECKKEYTIME.

- PartialSubject

соответствует флагу FLAG_FIND_SUBJECTISPARTIAL.

- IgnoreCached

соответствует флагу FLAG_FIND_IGNORESTORECACHE.

- IgnoreLocal

соответствует флагу FLAG_FIND_IGNORESTORELOCAL.

- SubjectAttribute

соответствует флагу FLAG_FIND_SUBJECTATTRIBUTE.

Класс **FindParameters** - (соответствует структуре *find_param_t*)

Параметры поиска сертификатов по заданному шаблону:

- свойство FindFlags **Flag**

Флаги поиска сертификатов.

- свойство Certificate **Template**

Шаблон сертификата для выполнения поиска.

CertificateCollection **FindCertificates**(FindParameters Param) - (соответствует функции *VCERT_FindCert*)

Метод поиска сертификатов по заданному шаблону

Аргументы:

- **Param** параметры поиска сертификатов по заданному шаблону.

Возвращаемые значения:

- найденные сертификаты.

Перечисление **EnumStoreCtx.EnumStoreFlag**

Флаги перечисления объектов:

- StorePersonal

соответствует флагу FLAG_ENUM_STORE_OBJECTS_STORE_PERS.

- StoreLocal

соответствует флагу FLAG_ENUM_STORE_OBJECTS_STORE_LOCL.

- StoreLdap

соответствует флагу FLAG_ENUM_STORE_OBJECTS_STORE_LDAP.

- StoreURI

соответствует флагу FLAG_ENUM_STORE_OBJECTS_STORE_SURI.

- ObjectCert

соответствует флагу FLAG_ENUM_STORE_OBJECTS_OBJECT_CER.

- ObjectCrl

соответствует флагу FLAG_ENUM_STORE_OBJECTS_OBJECT_CRL.

Класс **EnumStoreCtx**

Контекст операции перечисления объектов:

- Конструктор **EnumStoreCtx**(EnumStoreFlag flag).

byte[] **EnumStoreObjects**(EnumStoreCtx enumStoreCtx, bool bClose) - (соот-

ветствует функции *VCERT_EnumStoreObjects*)

Метод перечисления объектов справочников

Аргументы:

- **enumStoreCtx** контекст перечисления объектов справочников;
- **bClose** признак прекращения процесса перечисления.

Возвращаемые значения:

- блок памяти с очередным объектом в DER-кодировке. Для полного перечисления объектов справочника функцию следует вызывать до тех пор, пока возвращаемое значение не равно **null**.

byte[] **EnumStoreObjectsEx**(EnumStoreCtx enumStoreCtx, string uri, string gri, bool bClose) - (соответствует функции *VCERT_EnumStoreObjectsEx*)

Расширенный метод перечисления объектов справочников

Аргументы:

- **enumStoreCtx** контекст перечисления объектов справочников;
- **uri** URI справочника для перечисления объектов;
- **gri** логическое выражение вида **(&(vdSubjName=*Иван*)(vdKeyId=12*))**, позволяющее ограничить просматриваемые в ССС контейнеры;
- **bClose** признак прекращения процесса перечисления.

Возвращаемые значения:

- блок памяти с очередным объектом в DER-кодировке. Для полного перечисления объектов справочника функцию следует вызывать до тех пор, пока возвращаемое значение не равно **null**.

1.7.13 Вычисление ЭП CMS-сообщений

Класс **CmsSignParameters** - (соответствует структуре *sign_param_t*)

Параметры вычисления ЭП CMS-сообщений:

- свойство bool **AddSigner** - добавлять сертификат подписанта - фактически, рабочий сертификат - в CMS-сообщение;
- свойство bool **SubjectKeyId** - использовать для идентификации сертификата подписанта данные дополнения "Идентификатор ключа владельца". Если данный флаг не установлен, то для идентификации используется пара Имя издателя/Серийный номер;
- свойство bool **Envelope** - не добавлять ЭП к подписанному CMS-сообщению, а обернуть ЭП существующее CMS-сообщение целиком. Данный флаг разрешено использовать только при формировании совмещенной ЭП;
- свойство bool **CAdESBES** - добавить в ЭП аутентифицируемые атрибуты в соответствии со спецификацией **CAdES-BES**. Установка данного флага приводит к добавлению дополнительного аутентифицируемого атрибута **ESS signing-certificate**.

Методы блочного вычисления совмещенной ЭП CMS-сообщений

byte[] **CmsBlkAttSignMem**(CmsSignParameters Param, byte[] Data) - (соответствует функции VCERT_CmsBlkAttSignMem)

Метод блочного вычисления совмещенной ЭП блока памяти

Аргументы:

- **Param** параметры вычисления ЭП CMS-сообщения;
- **Data** данные (не нулевой длины) для вычисления ЭП.

Возвращаемые значения:

- подписанное CMS-сообщение.

void **CmsBlkAttSignFile**(CmsSignParameters Param, string Data, string Ocms) - (соответствует функции VCERT_CmsBlkAttSignFile)

Метод блочного вычисления совмещенной ЭП файла

Аргументы:

- **Param** параметры вычисления ЭП CMS-сообщений;
- **Data** файл (не нулевой длины) с данными для вычисления ЭП;
- **Ocms** файл с подписанным CMS-сообщением.

Методы потокового вычисления совмещенной ЭП CMS-сообщений

Класс **CmsStrAttSignCtx**

Контекст потоковой присоединенной подписи:

- конструктор **CmsStrAttSignCtx**();
- конструктор **CmsStrAttSignCtx**(CmsSignParameters signpar);
- свойство CmsSignParameters **signParam** параметры вычисления ЭП CMS-сообщений.

byte[] **CmsStrAttSignMem**(CmsStrAttSignCtx signContext, byte[] Data, bool IsLast) - (соответствует функциям VCERT_CmsStrAttSignInitMem, VCERT_CmsStrAttSignUpdateMem и VCERT_CmsStrAttSignFinalMem)

Метод потокового вычисления совмещенной ЭП блока памяти

Аргументы:

- **signContext** контекст потоковой присоединенной подписи;
- **Data** блок памяти с данными для вычисления ЭП (должно выполняться условие **Data.Length > 0**). При добавлении ЭП к подписанному CMS-сообщению должно выполняться условие **Data.Length ≥ 128**;
- **IsLast** признак завершения процесса подписи.

Возвращаемые значения:

- блок памяти с подписанным CMS-сообщением.

void **CmsStrAttSignFile**(CmsSignParameters Param, string Data, string Ocms) - (соответствует функции VCERT_CmsStrAttSignFile)

Метод потокового вычисления совмещенной ЭП файла**Аргументы:**

- **Param** параметры вычисления ЭП CMS-сообщений;
- **Data** файл (не нулевой длины) с данными для вычисления ЭП;
- **Ocms** файл с подписанным CMS-сообщением.

Методы блочного вычисления отдельной ЭП CMS-сообщений

byte[] **CmsBlkDetSignMem**(CmsSignParameters Param, byte[] Data, byte[] Icms) - (соответствует функции VCERT_CmsBlkDetSignMem)

Метод блочного вычисления отдельной ЭП блока памяти**Аргументы:**

- **Param** параметры вычисления ЭП CMS-сообщений;
- **Data** блок памяти (не нулевой длины) с данными для вычисления ЭП;
- **Icms** блок памяти (опциональный) с подписанным CMS-сообщением для добавления ЭП.

Возвращаемые значения:

- блок памяти с подписанным CMS-сообщением.

void **CmsBlkDetSignFile**(CmsSignParameters Param, string Data, string Icms, string Ocms) - (соответствует функции VCERT_CmsBlkDetSignFile)

Метод блочного вычисления отдельной ЭП файла**Аргументы:**

- **Param** параметры вычисления ЭП CMS-сообщений;
- **Data** файл (не нулевой длины) с данными для вычисления ЭП;
- **Icms** файл (опциональный) с подписанным CMS-сообщением для добавления ЭП;
- **Ocms** файл с подписанным CMS-сообщением.

Методы потокового вычисления отдельной ЭП CMS-сообщений**Класс CmsStrDetSignCtx**Контекст потоковой отдельной подписи:

- конструктор **CmsStrDetSignCtx**();
- конструктор **CmsStrDetSignCtx**(CmsSignParameters signpar, byte[] inCms);
- свойство CmsSignParameters **signParam** параметры вычисления ЭП CMS-сообщений.
- свойство byte[] **icms** входящее CMS сообщение (в случае добавления ЭП).

byte[] **CmsStrDetSignMem**(CmsStrDetSignCtx signContext, byte[] Data, bool IsLast) - (соответствует функциям VCERT_CmsStrDetSignInitMem, VCERT_

CmsStrDetSignUpdateMem и *VCERT_CmsStrDetSignFinalMem*)

Метод потокового вычисления отдельной ЭП блока памяти

Аргументы:

- ***signContext*** контекст потоковой отдельной подписи;
- ***Data*** блок памяти с продолжением данных для вычисления ЭП;
- ***IsLast*** признак завершения процесса подписи.

Возвращаемые значения:

- блок памяти с подписанным CMS-сообщением.

void **CmsStrDetSignFile**(CmsSignParameters Param, string Data, string Icms, string Ocms) - (соответствует функции *VCERT_CmsStrDetSignFile*)

Метод потокового вычисления отдельной ЭП файла

Аргументы:

- ***Param*** параметры вычисления ЭП CMS-сообщений;
- ***Data*** файл (не нулевой длины) с данными для вычисления ЭП;
- ***Icms*** файл (опциональный) с подписанным CMS-сообщением для добавления ЭП;
- ***Ocms*** файл с подписанным CMS-сообщением.

1.7.14 Проверка ЭП CMS-сообщений

Перечисление **CmsVerifyParameters.CmsVerifyFlags**

Флаги проверки ЭП CMS-сообщений:

- DeleteSignatures
соответствует флагу *FLAG_CMS_VERIFY_DELETESIGNATURES*.
- SignerKeyUsages
соответствует флагу *FLAG_CMS_VERIFY_SIGNERKEYUSAGES*.
- SignerPolicies
соответствует флагу *FLAG_CMS_VERIFY_SIGNERPOLICIES*.
- SignerExtKeyUsages
соответствует флагу *FLAG_CMS_VERIFY_SIGNEREXTKEYUSAGES*.
- MinimumSignatures
соответствует флагу *FLAG_CMS_VERIFY_MINIMUMSIGNATURES*.
- DoNotCheckTimes
соответствует флагу *FLAG_CMS_VERIFY_DONOTCHECKTIMES*.
- DoNotAddSigner
соответствует флагу *FLAG_CMS_VERIFY_DONOTADDSIGNER*.
- IgnoreAttachedSigner
соответствует флагу *FLAG_CMS_VERIFY_IGNOREATTACHEDSIGNER*.
- UseRevocationTime
соответствует флагу *FLAG_CMS_VERIFY_USEREVOCATIONTIME*.

- DoNotAddAttachedSigner
соответствует флагу *FLAG_CMS_VERIFY_DONOTADDATTACHEDSIGNER*.
- RequireAttachedSigner
соответствует флагу *FLAG_CMS_VERIFY_REQUIREATTACHEDSIGNER*.
- UseAttachedChain
соответствует флагу *FLAG_CMS_VERIFY_USEATTACHEDCHAIN*.
- RequiureAttachedChain
соответствует флагу *FLAG_CMS_VERIFY_REQUIREATTACHEDCHAIN*.

Класс **CmsVerifyParameters** - (соответствует структуре *verify_param_t*)
Параметры проверки ЭП CMS-сообщений:

- свойство *CmsVerifyFlags* **Flag**
Флаги проверки ЭП CMS-сообщений.
- свойство *KeyUsages* **KeyUsage**
Разрешенные области использования открытого ключа.
- метод *void* **AddPolicy** (*string* pol)
Добавить политику использования сертификата для проверки.
- метод *void* **AddExtendedKeyUsage** (*string* eku)
Добавить расширенную область применения ключа для проверки.
- свойство *int* **SignToDelete**
Количество ЭП, которые необходимо удалить, начиная с конца CMS-сообщения. Для удаления всех ЭП следует подать значение, равное **-1**. При проверке совмещенных потоковых ЭП разрешено подавать только значения **0** и **-1**.
- свойство *int* **MinimumSignNumber**
Минимально допустимое количество ЭП в CMS-сообщении.

Класс **VerifyResult** - (соответствует структуре *sign_status_t*)
Результат проверки конкретной ЭП CMS-сообщения:

- свойство *uint* **RawStatus**
Результат проверки конкретной ЭП:
 - **== 0** - ЭП проверена успешно;
 - **!= 0** - код ошибки проверки ЭП.
- свойство *string* **Description**
Описание результата проверки.
- свойство *DateTime* **Time**
Время вычисления ЭП в часовом поясе UTC, взятое из аутентифицируемого атрибута **rsaSigningTime** данной ЭП (равно **0** в случае отсутствия указанного атрибута в ЭП).
- свойство *Certificate* **SignerCertificate**
Сертификат, на котором выполнялась проверка ЭП, т.е. сертификат отправителя данной ЭП.

Методы блочной проверки совмещенных ЭП CMS-сообщений

bool **CmsBlkAttVerifyMem**(CmsVerifyParameters Param, byte[] Icms, out byte[] Data, out VerifyResultCollection VerifyResult) - (соответствует функции VCERT_CmsBlkAttVerifyMem)

Метод блочной проверки совмещенных ЭП блока памяти

Аргументы:

- **Param** параметры проверки ЭП CMS-сообщений;
- **Icms** блок памяти (не нулевой длины) с подписанным CMS-сообщением;
- **Data** блок памяти с выходными данными. Заполняется при установке флага проверки **DeleteSignatures** (в случае успешной проверки всех ЭП CMS-сообщения), иначе равен **null**;
- **VerifyResult** результат проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **true** в случае успешной проверки всех ЭП (необходимо анализировать **VerifyResult**);
- **false** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать **VerifyResult**);
- генерирует **VcertException** в случае другой ошибки (запрещено анализировать **VerifyResult**).

bool **CmsBlkAttVerifyFile**(CmsVerifyParameters Param, string Icms, string Data, out VerifyResultCollection VerifyResult) - (соответствует функции VCERT_CmsBlkAttVerifyFile)

Метод блочной проверки совмещенных ЭП файла

Аргументы:

- **Param** параметры проверки ЭП CMS-сообщений;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением;
- **Data** файл с выходными данными. Создается при установке флага проверки **DeleteSignatures** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **null**;
- **VerifyResult** результат проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **true** в случае успешной проверки всех ЭП (необходимо анализировать **VerifyResult**);
- **false** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать **VerifyResult**);
- генерирует **VcertException** в случае другой ошибки (запрещено анализировать **VerifyResult**).

Методы потоковой проверки совмещенных ЭП CMS-сообщений

Класс **CmsStrAttVerifyCtx**

Контекст потоковой проверки совмещенных ЭП CMS-сообщений:

- конструктор **CmsStrAttVerifyCtx()**;
- конструктор **CmsStrAttVerifyCtx(CmsVerifyParameters verifyParam)**;
- свойство **CmsVerifyParameters verifyParam**
параметры проверки ЭП CMS-сообщений.

bool CmsStrAttVerifyMem(CmsStrAttVerifyCtx verifyContext, byte[] Icms, bool IsLast, out byte[] Data, out VerifyResultCollection VerifyResult) - (*соответствует функциям VCERT_CmsStrAttVerifyInitMem, VCERT_CmsStrAttVerifyUpdateMem и VCERT_CmsStrAttVerifyFinalMem*)

Метод потоковой проверки совмещенных ЭП блока памяти

Аргументы:

- **verifyContext** контекст потоковой операции;
- **Icms** блок памяти (не нулевой длины) с очередной частью подписанного CMS-сообщения;
- **IsLast** признак завершения процесса подписи;
- **Data** блок памяти с выходными данными. Заполняется при установке флага проверки **DeleteSignatures**, иначе равен **null**;
- **VerifyResult** результат проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **true** в случае успешной проверки всех ЭП (необходимо анализировать **VerifyResult**);
- **false** если проверка не завершена, при завершении - в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать **VerifyResult**);
- генерирует **VcertException** в случае другой ошибки (запрещено анализировать **VerifyResult**).

bool CmsStrAttVerifyFile(CmsVerifyParameters Param, string Icms, string Data, out VerifyResultCollection VerifyResult) - (*соответствует функции VCERT_CmsStrAttVerifyFile*)

Метод потоковой проверки совмещенных ЭП файла

Аргументы:

- **VerifyParam** параметров проверки ЭП CMS-сообщений;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением;
- **Data** файл с выходными данными. Создается при установке флага проверки **DeleteSignatures** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **null**;
- **VerifyResult** результат проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **true** в случае успешной проверки всех ЭП (необходимо анализировать

VerifyResult);

- **false** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать **VerifyResult**);
- генерирует **VcertException** в случае другой ошибки (запрещено анализировать **VerifyResult**).

Методы блочной проверки отделенных ЭП CMS-сообщений

bool **CmsBlkDetVerifyMem**(CmsVerifyParameters Param, byte[] Data, byte[] Icms, out byte[] Ocms, out VerifyResultCollection VerifyResult) - (*соответствует функции VCERT_CmsBlkDetVerifyMem*)

Метод блочной проверки отделенных ЭП блока памяти

Аргументы:

- **Param** параметры проверки ЭП CMS-сообщений;
- **Data** блок памяти (не нулевой длины) с данными для проверки ЭП;
- **Icms** блок памяти (не нулевой длины) с подписанным CMS-сообщением;
- **Ocms** блок памяти с выходным CMS-сообщением. Заполняется при установке флага проверки **DeleteSignatures** (в случае успешной проверки всех ЭП CMS-сообщения), иначе равен **null**;
- **VerifyResult** результат проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **true** в случае успешной проверки всех ЭП (необходимо анализировать **VerifyResult**);
- **false** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать **VerifyResult**);
- генерирует **VcertException** в случае другой ошибки (запрещено анализировать **VerifyResult**).

bool **CmsBlkDetVerifyFile**(CmsVerifyParameters Param, string Data, string Icms, string Ocms, out VerifyResultCollection VerifyResult) - (*соответствует функции VCERT_CmsBlkDetVerifyFile*)

Метод блочной проверки отделенных ЭП файла

Аргументы:

- **Param** параметры проверки ЭП CMS-сообщений;
- **Data** файл (не нулевой длины) с данными для проверки ЭП;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением;
- **Ocms** файл с выходным CMS-сообщением. Создается при установке флага проверки **DeleteSignatures** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **null**;
- **VerifyResult** результат проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **true** в случае успешной проверки всех ЭП (необходимо анализировать

VerifyResult);

- **false** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать **VerifyResult**);
- генерирует **VcertException** в случае другой ошибки (запрещено анализировать **VerifyResult**).

Функции потоковой проверки отделенных ЭП CMS-сообщений

Класс **CmsStrDetVerifyCtx**

Контекст потоковой проверки отделенных ЭП CMS-сообщений:

- конструктор **CmsStrDetVerifyCtx**(byte[] Icms);
- конструктор **CmsStrDetVerifyCtx**(byte[] Icms, CmsVerifyParameters verifyParam);
- свойство CmsVerifyParameters **verifyParam**
параметры проверки ЭП CMS-сообщений.
- свойство byte[] **icms**
входящее CMS сообщение в виде блока данных.

bool **CmsStrDetVerifyMem**(CmsStrDetVerifyCtx verifyContext, byte[] Data, bool IsLast, out byte[] Ocms, out VerifyResultCollection VerifyResult) - (*соответствует функциям VCERT_CmsStrDetVerifyInitMem, VCERT_CmsStrDetVerifyUpdateMem и VCERT_CmsStrDetVerifyFinalMem*)

Метод потоковой проверки отделенных ЭП блока памяти

Аргументы:

- **verifyContext** контекст потоковой операции;
- **Data** блок памяти с подписанным CMS-сообщением;
- **IsLast** признак завершения процесса проверки;
- **Ocms** блок памяти с выходным CMS-сообщением. Заполняется при установке флага проверки **DeleteSignatures** (в случае успешной проверки всех ЭП CMS-сообщения), иначе равен **null**;
- **VerifyResult** результат проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **true** в случае успешной проверки всех ЭП (необходимо анализировать **VerifyResult**);
- **false** если проверка не завершена, при завершении - в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать **VerifyResult**);
- генерирует **VcertException** в случае другой ошибки (запрещено анализировать **VerifyResult**).

bool **CmsStrDetVerifyFile**(CmsVerifyParameters Param, string Data, string Icms, string Ocms, out VerifyResultCollection VerifyResult) - (*соответствует функции VCERT_CmsStrDetVerifyFile*)

Метод потоковой проверки отделенных ЭП файла

Аргументы:

- **Param** параметры проверки ЭП CMS-сообщений;
- **Data** файл (не нулевой длины) с данными для проверки ЭП;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением;
- **Ocms** файл с выходным CMS-сообщением. Создается при установке фла-га проверки **DeleteSignatures** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **null**;
- **VerifyResult** результат проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **true** в случае успешной проверки всех ЭП (необходимо анализировать **VerifyResult**);
- **false** в случае ошибки проверки хотя бы одной ЭП (необходимо анализи-ровать **VerifyResult**);
- генерирует **VcertException** в случае другой ошибки (запрещено анализи-ровать **VerifyResult**).

1.7.15 Зашифрование CMS-сообщенийПеречисление **CmsEncryptParameters.CmsEncryptFlags**Флаги зашифрования CMS-сообщений:

- UseRemoteSearch
соответствует флагу *FLAG_CMS_ENCRYPT_USEREMOTESEARCH*.
- DoNotCheckTimes
соответствует флагу *FLAG_CMS_ENCRYPT_DONOTCHECKTIMES*.
- DoNotVerifyChain
соответствует флагу *FLAG_CMS_ENCRYPT_DONOTVERIFYCHAIN*.
- AddRemoteToLocal
соответствует флагу *FLAG_CMS_ENCRYPT_ADDREMOTETOLOCAL*.
- DoNotCheckKeyTime
соответствует флагу *FLAG_CMS_ENCRYPT_DONOTCHECKKEYTIME*.
- SubjectIsPartial
соответствует флагу *FLAG_CMS_ENCRYPT_SUBJECTISPARTIAL*.
- IgnoreStoreCache
соответствует флагу *FLAG_CMS_ENCRYPT_IGNORESTORECACHE*.
- IgnoreStoreLocal
соответствует флагу *FLAG_CMS_ENCRYPT_IGNORESTORELOCAL*.
- SubjectAttribute
соответствует флагу *FLAG_CMS_ENCRYPT_SUBJECTATTRIBUTE*.
- SubjectKeyId
соответствует флагу *FLAG_CMS_ENCRYPT_SUBJKEYID*.
- GostR34_12_15MG
соответствует флагу *FLAG_CMS_ENCRYPT_GOST_R_34_12_15MG*.

- GostR34_12_15GH
соответствует флагу FLAG_CMS_ENCRYPT_GOST_R_34_12_15GH.
- AddOmacAttribute
соответствует флагу FLAG_CMS_ENCRYPT_ADDOMACATTRIBUTE.
- RecipientKeyAgreement
соответствует флагу FLAG_CMS_ENCRYPT_RECIPKEYAGREEMENT.

Класс **CmsEncryptParameters** - (*соответствует структуре encrypt_param_t*)

Параметры зашифрования CMS-сообщений:

- свойство CmsEncryptFlags **Flag**
Флаги зашифрования CMS-сообщений.
- метод void **AddReceiver**(Certificate cert)
Добавление шаблона сертификата получателя.

Методы блочного зашифрования CMS-сообщений

byte[] **CmsBlkEncryptMem**(CmsEncryptParameters Param, byte[] Data) - (*соответствует функции VCERT_CmsBlkEncryptMem*)

Метод блочного зашифрования блока памяти

Аргументы:

- **Param** параметры зашифрования CMS-сообщений;
- **Data** блок памяти (не нулевой длины) с данными для зашифрования.

Возвращаемые значения:

- зашифрованное CMS-сообщение.

void **CmsBlkEncryptFile**(CmsEncryptParameters Param, string Data, string Ocms) - (*соответствует функции VCERT_CmsBlkEncryptFile*)

Метод блочного зашифрования файла

Аргументы:

- **Param** параметры зашифрования CMS-сообщений;
- **Data** файл (не нулевой длины) с данными для зашифрования;
- **Ocms** файл с зашифрованным CMS-сообщением.

Методы потокового зашифрования CMS-сообщений

Класс **CmsStrEncryptCtx**

Контекст потокового зашифрования CMS-сообщений:

- конструктор **CmsStrEncryptCtx**();
- конструктор **CmsStrEncryptCtx**(CmsEncryptParameters encryptParam);
- свойство CmsEncryptParameters **encryptParam**
параметры зашифрования CMS-сообщений.

byte[] **CmsStrEncryptMem**(CmsStrEncryptCtx strEncryptContext, byte[] Data, bool IsLast) - (соответствует функциям VCERT_CmsStrEncryptInitMem, VCERT_CmsStrEncryptUpdateMem и VCERT_CmsStrEncryptFinalMem)

Метод потокового зашифрования блока памяти

Аргументы:

- **strEncryptContext** контекст потокового зашифрования;
- **Data** блок памяти с данными для зашифрования;
- **IsLast** признак завершения процесса зашифрования.

Возвращаемые значения:

- блок памяти с зашифрованным CMS-сообщением.

void **CmsStrEncryptFile**(CmsEncryptParameters Param, string Data, string Ocms) - (соответствует функции VCERT_CmsStrEncryptFile)

Метод потокового зашифрования файла

Аргументы:

- **Param** параметры зашифрования CMS-сообщений;
- **Data** файл (не нулевой длины) с данными для зашифрования;
- **Ocms** файл с зашифрованным CMS-сообщением.

1.7.16 Расшифрование CMS-сообщений

Класс **CmsDecryptParameters** - (соответствует структуре decrypt_param_t)

Параметры расшифрования CMS-сообщений:

- свойство CmsDecryptFlags **Flag** (зарезервировано, должно быть равно 0)
- Флаги расшифрования CMS-сообщений.*

Методы блочного расшифрования CMS-сообщений

byte[] **CmsBlkDecryptMem**(CmsDecryptParameters Param, byte[] Icms) - (соответствует функции VCERT_CmsBlkDecryptMem)

Метод блочного расшифрования блока памяти

Аргументы:

- **Param** параметры расшифрования CMS-сообщений;
- **Icms** блок памяти (не нулевой длины) с зашифрованным CMS-сообщением.

Возвращаемые значения:

- блок памяти с расшифрованными данными.

void **CmsBlkDecryptFile**(CmsDecryptParameters Param, string Icms, string Data) - (соответствует функции VCERT_CmsBlkDecryptFile)

Метод блочного расшифрования файла

Аргументы:

- **Param** параметры расшифрования CMS-сообщений;

- **Icms** файл (не нулевой длины) с зашифрованным CMS-сообщением;
- **Data** файл с расшифрованными данными.

Методы потокового расшифрования CMS-сообщений

Класс **CmsStrDecryptCtx**

Контекст потокового расшифрования CMS-сообщений:

- конструктор **CmsStrDecryptCtx()**;
- свойство CmsDecryptParameters **decryptParam**
параметры расшифрования CMS-сообщений.

byte[] **CmsStrDecryptMem**(CmsStrDecryptCtx strDecryptCtx, byte[] Icms, bool IsLast) - (*соответствует функциям VCERT_CmsStrDecryptInitMem, VCERT_CmsStrDecryptUpdateMem и VCERT_CmsStrDecryptFinalMem*)

Метод потокового расшифрования блока памяти

Аргументы:

- **strDecryptCtx** контекст потокового расшифрования;
- **Icms** блок памяти с частью зашифрованного CMS-сообщения (должно выполняться условие **Icms.Length** ≥ 128);
- **IsLast** признак завершения процесса расшифрования.

Возвращаемые значения:

- блок памяти с частью расшифрованных данных.

void **CmsStrDecryptFile**(CmsDecryptParameters Param, string Icms, string Data) - (*соответствует функции VCERT_CmsStrDecryptFile*)

Метод потокового расшифрования файла

Аргументы:

- **Param** параметры расшифрования CMS-сообщений;
- **Icms** файл (не нулевой длины) с зашифрованным CMS-сообщением;
- **Data** файл с расшифрованными данными.

1.7.17 Преобразование совмещенных и отделенных ЭП

Методы блочного преобразования отделенных ЭП в совмещенные

byte[] **CmsBlkSignAttachMem**(byte[] Data, byte[] Icms, SignAttachFlags flag) - (*соответствует функции VCERT_CmsBlkSignAttachMem*)

Метод блочного преобразования отделенных ЭП блока памяти в совмещенные

Аргументы:

- **Data** блок памяти (не нулевой длины) с подписанными данными;
- **Icms** блок памяти (не нулевой длины) с подписанным CMS-сообщением с отделенными ЭП;
- **flag** флаги преобразования (зарезервировано, должно быть равно **SignAttachFlags.NoFlag**).

Возвращаемые значения:

- блок памяти с подписанным CMS-сообщением с совмещенными ЭП.

void **CmsBlkSignAttachFile**(string Data, string Icms, string Ocms, SignAttachFlags flag) - *(соответствует функции VCERT_CmsBlkSignAttachFile)*
Метод блочного преобразования отделенных ЭП файла в совмещенные

Аргументы:

- **Data** файл (не нулевой длины) подписанными с данными;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением с отделенными ЭП;
- **Ocms** файл с подписанным CMS-сообщением с совмещенными ЭП;
- **flag** флаги преобразования (зарезервировано, должно быть равно **SignAttachFlags.NoFlag**).

Методы блочного преобразования совмещенных ЭП в отделенные

void **CmsBlkSignDetachMem**(byte[] Icms, out byte[] Data, out byte[] Ocms, SignDetachFlags flag) - *(соответствует функции VCERT_CmsBlkSignDetachMem)*
Метод блочного преобразования совмещенных ЭП блока памяти в отделенные

Аргументы:

- **Icms** блок памяти (не нулевой длины) с подписанным CMS-сообщением с совмещенными ЭП;
- **Data** блок памяти с подписанными данными;
- **Ocms** блок памяти с подписанным CMS-сообщением с отделенными ЭП;
- **flag** флаги преобразования (зарезервировано, должно быть равно **SignDetachFlags.NoFlag**).

void **CmsBlkSignDetachFile**(string Icms, string Data, string Ocms, SignDetachFlags flag) - *(соответствует функции VCERT_CmsBlkSignDetachFile)*
Метод блочного преобразования совмещенных ЭП файла в отделенные

Аргументы:

- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением с совмещенными ЭП;
- **Data** файл с подписанными данными;
- **Ocms** файл с подписанным CMS-сообщением с отделенными ЭП;
- **flag** флаги преобразования (зарезервировано, должно быть равно **SignDetachFlags.NoFlag**).

Методы потокового преобразования совмещенных ЭП в отделенные**Класс CmsStrSignDetachCtx**

Контекст потокового преобразования совмещенных ЭП в отделенные:

- конструктор **CmsStrDecryptCtx()**;
- конструктор **CmsStrSignDetachCtx**(VcertObject.SignDetachFlags flags).

void **CmsStrSignDetachMem**(CmsStrSignDetachCtx signDetachContext, byte[] Icms, bool IsLast, out byte[] Data, out byte[] Ocms) - (соответствует функциям *VCERT_CmsStrSignDetachInitMem*, *VCERT_CmsStrSignDetachUpdateMem* и *VCERT_CmsStrSignDetachFinalMem*)

Метод потокового преобразования совмещенных ЭП блока памяти

Аргументы:

- **signDetachContext** контекст потоковой операции;
- **Icms** блок памяти с частью подписанного CMS-сообщения с совмещенными ЭП (должно выполняться условие **Icms.Length** \geq **128**);
- **IsLast** признак завершения процесса преобразования;
- **Data** блок памяти с частью подписанных данных;
- **Ocms** блок памяти с подписанным CMS-сообщением с отделенными ЭП.

void **CmsStrSignDetachFile**(string Icms, string Data, string Ocms, SignDetachFlags flag) - (соответствует функции *VCERT_CmsStrSignDetachFile*)

Метод потокового преобразования совмещенных ЭП файла в отделенные

Аргументы:

- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением с совмещенными ЭП;
- **Data** файл с подписанными данными;
- **Ocms** файл с подписанным CMS-сообщением с отделенными ЭП;
- **flag** флаги преобразования (зарезервировано, должно быть равно **SignDetachFlags.NoFlag**).

1.7.18 Получение информации о CMS-сообщениях

Перечисление **CmsCertIdType**

Тип идентификатора сертификатов CMS-сообщения:

- Issn

соответствует флагу *FLAG_CMS_CERTID_TYPE_ISSN*.

- Skid

соответствует флагу *FLAG_CMS_CERTID_TYPE_SKID*.

Класс **CmsCertId** - (соответствует структуре *cms_certid_t*)

Идентификатор сертификата CMS-сообщения:

- CmsCertIdType **Type**

Тип идентификатора сертификатов CMS-сообщения.

- string **Issuer**

Поле с X.500-именем издателя сертификата в виде строки, например *CN=TestUser,DC=X509,DC=RU* (заполняется для типа структуры идентифи-

катора **Issn**).

– string **SerialNum**

Поле с серийным номером сертификата в виде строки, например 00:01:02:03:04:05:06:07:08:09:0A:0B:0C:0D:0E:0F (заполняется для типа структуры идентификатора **Issn**).

– string **CertHash**

Поле с хэш-значением пары Имя издателя/Серийный номер сертификата в виде строки, например 00:01:02:03:04:05:06:07:08:09:0A:0B:0C:0D:0E:0F (заполняется для типа структуры идентификатора **Issn**).

– string **SubjectKeyId**

Поле с данными дополнения "Идентификатор ключа владельца" в виде строки, например 00:01:02:03:04:05:06:07:08:09:0A:0B:0C:0D:0E:0F (заполняется для типа структуры идентификатора **Skid**).

Класс **CmsSignerInfo** - (соответствует структуре cms_signinf_t)

Информация о подписанте CMS-сообщения:

– uint **Version**

Номер версии структуры информации о подписанте CMS-сообщения.

– string **DigestAlg**

Строка объектного идентификатора (OID) алгоритма хэширования.

– string **SignatureAlg**

Строка объектного идентификатора (OID) алгоритма ЭП.

– CmsCertId **SignerId**

Идентификатор сертификата подписанта CMS-сообщения.

– DateTime **SigningTime**

Поле со временем вычисления ЭП, взятым из аутентифицированного атрибута **PKCS#9 signingTime** (при отсутствии данного атрибута значение равно **DateTime.MinValue**).

– string[] **AuthAttrs**

Массив строк объектных идентификаторов аутентифицированных атрибутов, находящихся в ЭП.

– string[] **UnauthAttrs**

Массив строк объектных идентификаторов неаутентифицированных атрибутов, находящихся в ЭП.

Перечисление **CmsRecipientInfoType**

Тип информации о получателе CMS-сообщения:

– KeyTransport

соответствует флагу **FLAG_CMS_RECINF_TYPE_KTRI**.

– KeyAgreement

соответствует флагу **FLAG_CMS_RECINF_TYPE_KARI**.

Класс **CmsRecipientInfo** - (соответствует структуре cms_recinf_t)

Информация о получателе CMS-сообщения:

- CmsRecipientInfoType **Type**

Поле с типом структуры информации о получателе CMS-сообщения.

- uint **Index**

*Номер (индекс) структуры информации о получателе CMS-сообщения (заполняется для типа информации о получателе **KeyAgreement**).*

- uint **Version**

Номер версии структуры информации о получателе CMS-сообщения.

- CmsCertId **OriginatorId**

Идентификатор сертификата отправителя CMS-сообщения (данное поле не заполняется для CMS-сообщений, зашифрованных анонимным способом).

- string **PublicKeyAlg**

*Строка объектного идентификатора (OID) алгоритма открытого ключа сертификата получателя (заполняется для типа структуры информации о получателе **KeyAgreement**).*

- string **CipherAlg**

Строка объектного идентификатора (OID) алгоритма согласования или зашифрования сеансового ключа.

- CmsCertId **RecipientId**

Идентификатор сертификата получателя CMS-сообщения.

Перечисление **CmsMsgInfoType**

Тип CMS-сообщения:

- Signed

соответствует флагу FLAG_CMS_MSGINF_TYPE_SIGN.

- Encrypted

соответствует флагу FLAG_CMS_MSGINF_TYPE_ENVL.

Перечисление **CmsMsgInfoFlags**

Флаги CMS-сообщения:

- Ndef

соответствует флагу FLAG_CMS_MSGINF_FLAG_NDEF.

- Detach

соответствует флагу FLAG_CMS_MSGINF_FLAG_DTCH.

Перечисление **CmsMsgInfo** - (соответствует структуре cms_msginf_t)

Информация о CMS-сообщении:

- CmsMsgInfoType **Type**

Тип CMS-сообщения.

- CmsMsgInfoFlags **Flags**

Флаги CMS-сообщения.

- CmsSignerInfo[] **Signers**

*Массив с информацией о подписантах CMS-сообщения (заполняется только для CMS-сообщений типа **Signed**).*

- CmsRecipientInfo[] **Recipients**

Массив с информацией о получателях CMS-сообщения (заполняется только для CMS-сообщений типа **Encrypted**).

- string **CipherAlg**

Строка объектного идентификатора (OID) алгоритма шифрования данных (заполняется только для CMS-сообщений типа **Encrypted**).

- string[] **UnprotAttrs**

Массив строк объектных идентификаторов незащищенных атрибутов (заполняется только для CMS-сообщений типа **Encrypted**).

Методы блочного получения информации о CMS-сообщениях

CmsMsgInfo **CmsBlkMsgInfMem**(byte[] Cms, MsgInfFlags Flag) - (соответствует функции VCERT_CmsBlkMsgInfMem)

Метод блочного получения информации о CMS-сообщениях в виде блока памяти

Аргументы:

- **Cms** (не нулевой длины) блок памяти с CMS-сообщением;
- **Flag** флаги (зарезервировано, должно быть равно **MsgInfFlags.NoFlag**).

Возвращаемые значения:

- информация о CMS-сообщении.

CmsMsgInfo **CmsBlkMsgInfFile**(string Cms, MsgInfFlags Flag) - (соответствует функции VCERT_CmsBlkMsgInfFile)

Метод блочного получения информации о CMS-сообщениях в виде файла

Аргументы:

- **Cms** файл (не нулевой длины) с CMS-сообщением;
- **Flag** флаги (зарезервировано, должно быть равно **MsgInfFlags.NoFlag**).

Возвращаемые значения:

- информация о CMS-сообщении.

Методы потокового получения информации о CMS-сообщениях

Класс **CmsStrMsgInfoCtx**

Контекст потокового получения информации о CMS сообщении:

- конструктор **CmsStrMsgInfoCtx()**.

CmsMsgInfo **CmsStrMsgInfMem**(CmsStrMsgInfoCtx strMsgInfoCtx, byte[] Cms, bool IsLast) - (соответствует функциям VCERT_CmsStrMsgInfInitMem, VCERT_CmsStrMsgInfUpdateMem и VCERT_CmsStrMsgInfFinalMem)

Метод потокового получения информации о блоке памяти CMS

Аргументы:

- **strMsgInfoCtx** контекст потоковой операции;

– **Cms** блок памяти с частью CMS-сообщения (должно выполняться условие **Cms.Length** \geq 128);

– **IsLast** признак завершения процесса преобразования.

Возвращаемые значения:

– информация о CMS-сообщении.

CmsMsgInfo CmsStrMsgInfFile(string Cms, MsgInfFlags Flag) - (соответствует функции VCERT_CmsStrMsgInfFile)

Метод потокового получения информации о CMS-сообщениях в виде файла

Аргументы:

– **Cms** файл (не нулевой длины) с CMS-сообщением;

– **Flag** флаги (зарезервировано, должно быть равно **MsgInfFlags.NoFlag**).

Возвращаемые значения:

– информация о CMS-сообщении.

1.7.19 Простановка и проверка штампов времени

Перечисление **TspRequestParameters.TspRequestFlags**

Флаги простановки штампа времени CMS-сообщений:

– AddNonce

соответствует флагу FLAG_TSP_REQUEST_INCLUDENONCE.

– CertRequest

соответствует флагу FLAG_TSP_REQUEST_ATTACHEDSIGNER.

Класс **TspRequestParameters** - (соответствует структуре *tsp_request_param_t*)

Параметры простановки штампа времени CMS-сообщений:

– TspRequestFlags **Flags**

Флаги простановки штампа времени CMS-сообщений.

– uint **Index**

Порядковый номер (индекс) ЭП CMS-сообщения, для которой следует запросить штамп времени.

Перечисление **TspResponseParameters.TspResponseFlags**

Флаги вычисления ЭП и формирования штампов времени:

– AddTsaName

соответствует флагу FLAG_TSP_RESPONSE_INCLUDETSANAME.

Класс **TspResponseParameters** - (соответствует структуре *tsp_response_param_t*)

Параметры вычисления ЭП и формирования штампов времени:

– TspResponseFlags **Flags**

Флаги вычисления ЭП и формирования штампов времени.

Перечисление **TspVerifyParameters.TspVerifyFlags**

Флаги проверки штампа времени CMS-сообщений:

– NoAttachedCert

соответствует флагу *FLAG_TSP_VERIFY_IGNOREATTACHEDSIGNER*.

Класс **TspVerifyParameters** - (соответствует структуре *tsp_verify_param_t*)

Параметры проверки штампа времени CMS-сообщений:

– TspVerifyFlags **Flag**

Флаги проверки штампа времени CMS-сообщений:

– uint **Index**

Поле с порядковым номером (индексом) ЭП CMS-сообщения, для которой следует проверить штамп времени.

Класс **TspVerifyResult** - (соответствует структуре *tsp_verify_result_t*)

Результат проверки штампа времени для заданной ЭП:

– DateTime **time**

Время простановки (вычисления ЭП) штампа времени в часовом поясе UTC.

– Certificate **cert**

Сертификат, на котором выполнялась проверка ЭП, т.е. сертификат сервера штампов времени (равно **null** в случае ненахождения сертификата сервера штампов времени).

Методы блочной простановки и проверки штампов времени

byte[] **TspBlkRequestFromCmsMem**(TspRequestParameters Param, byte[] Icms) - (соответствует функции *VCERT_TspBlkRequestFromCmsMem*)

Метод блочного создания запроса на получение штампа времени блока памяти

Аргументы:

– **Param** параметры простановки штампа времени CMS-сообщений;

– **Icms** блок памяти (не нулевой длины) с подписанным CMS-сообщением.

Возвращаемые значения:

– блок памяти с запросом на получение штампа времени.

byte[] **TspBlkRequestFromCmsFile**(TspRequestParameters Param, string Icms) - (соответствует функции *VCERT_TspBlkRequestFromCmsFile*)

Метод блочного создания запроса на получение штампа времени файла

Аргументы:

– **Param** параметры простановки штампа времени CMS-сообщений;

– **Icms** файл (не нулевой длины) с подписанным CMS-сообщением.

Возвращаемые значения:

- блок памяти с запросом на получение штампа времени.

byte[] **TspSignResponse**(TspResponseParameters Param, byte[] Request) - (соответствует функции VCERT_TspSignResponse)

Метод блочного вычисления ЭП и формирования штампов времени

Аргументы:

- **Param** параметры вычисления ЭП и формирования штампов времени;
- **Request** блок памяти (не нулевой длины) с запросом на получение штампа времени.

Возвращаемые значения:

- блок памяти с сформированным штампом времени.

TspVerifyResult **TspVerifyResponse**(TspVerifyParameters Param, byte[] Response) - (соответствует функции VCERT_TspVerifyResponse)

Метод блочной проверки ЭП сформированного штампа времени

Аргументы:

- **Param** параметры проверки штампа времени CMS-сообщений;
- **Response** блок памяти (не нулевой длины) с сформированным штампом времени.

Возвращаемые значения:

- результат проверки штампа времени для заданной ЭП.

byte[] **TspBlkUrlStampCmsMem**(TspRequestParameters Param, string Url, byte[] Icms) - (соответствует функции VCERT_TspBlkUrlStampCmsMem)

Метод блочной простановки штампа времени блока памяти CMS на сервере

Аргументы:

- **Param** параметры простановки штампа времени CMS-сообщений;
- **Url** строка (URI) с адресом сервера штампов времени;
- **Icms** блок памяти (не нулевой длины) с подписанным CMS-сообщением.

Возвращаемые значения:

- блок памяти с подписанным CMS-сообщением, включающим штамп времени.

void **TspBlkUrlStampCmsFile**(TspRequestParameters Param, string Url, string Icms, string Ocms) - (соответствует функции VCERT_TspBlkUrlStampCmsFile)

Метод блочной простановки штампа времени файла CMS на сервере

Аргументы:

- **Param** параметры простановки штампа времени CMS-сообщений;
- **Url** строка (URI) с адресом сервера штампов времени;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением;
- **Ocms** файл с подписанным CMS-сообщением, включающим штамп време-

ни.

TspVerifyResult **TspBlkVerifyCmsMem**(TspVerifyParameters Param, byte[] Icms) - *(соответствует функции VCERT_TspBlkVerifyCmsMem)*
Метод блочной проверки штампа времени блока памяти CMS

Аргументы:

- **Param** параметры проверки штампа времени CMS-сообщений;
- **Icms** блок памяти (не нулевой длины) с подписанным CMS-сообщением, включающим штамп времени.

Возвращаемые значения:

- результат проверки штампа времени для заданной ЭП.

TspVerifyResult **TspBlkVerifyCmsFile**(TspVerifyParameters Param, string Icms) - *(соответствует функции VCERT_TspBlkVerifyCmsFile)*
Метод блочной проверки штампа времени файла CMS

Аргументы:

- **Param** параметры проверки штампа времени CMS-сообщений;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением, включающим штамп времени.

Возвращаемые значения:

- результат проверки штампа времени для заданной ЭП.

Методы потоковой простановки и проверки штампов времени

Класс **CmsStrUrlStampMemCtx**

Контекст потоковой простановки штампа времени CMS на сервере:

- конструктор **CmsStrUrlStampMemCtx**(string url);
- конструктор **CmsStrUrlStampMemCtx**(TspRequestParameters reqParam, string url);
- свойство TspRequestParameters **requestParam**

Параметры простановки штампа времени CMS-сообщений.

byte[] **TspStrUrlStampCmsMem**(CmsStrUrlStampMemCtx StrUrlStampCtx, byte[] Icms, bool IsLast) - *(соответствует функциям VCERT_TspStrUrlStampCmsInitMem, VCERT_TspStrUrlStampCmsUpdateMem и VCERT_TspStrUrlStampCmsFinalMem)*

Метод потоковой простановки штампа времени CMS на сервере

Аргументы:

- **StrUrlStampCtx** контекст потоковой операции;
- **Icms** блок памяти с частью подписанного CMS-сообщения (должно выполняться условие **Icms.Length** ≥ 128);
- **IsLast** признак завершения процесса простановки штампа времени.

Возвращаемые значения:

– блок памяти с частью подписанного CMS-сообщения, включающего штамп времени.

`void TspStrUrlStampCmsFile(TspRequestParameters Param, string Url, string Icms, string Ocms)` - (соответствует функции *VCERT_TspStrUrlStampCmsFile*)
Метод потоковой простановки штампа времени файла CMS на сервере

Аргументы:

- **Param** параметры простановки штампа времени CMS-сообщений;
- **Url** строка (URI) с адресом сервера штампов времени;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением;
- **Ocms** файл с подписанным CMS-сообщением, включающим штамп времени.

Класс CmsStrTspVerifyMemCtx

Контекст потоковой проверки штампа времени блока памяти CMS:

- конструктор **CmsStrTspVerifyMemCtx()**;
- свойство `TspVerifyParameters` **verifyParam**

Параметры проверки штампа времени CMS-сообщений.

`TspVerifyResult TspStrVerifyCmsMem(CmsStrTspVerifyMemCtx strTspVerifyCtx, byte[] Icms, bool IsLast)` - (соответствует функциям *VCERT_TspStrVerifyCmsInitMem*, *VCERT_TspStrVerifyCmsUpdateMem* и *VCERT_TspStrVerifyCmsFinalMem*)

Метод потоковой проверки штампа времени блока памяти CMS

Аргументы:

- **strTspVerifyCtx** контекст потоковой операции;
- **Icms** блок памяти с частью подписанного CMS-сообщения, включающего штамп времени (должно выполняться условие **Icms.Length** ≥ 128);
- **IsLast** признак завершения процесса проверки штампа времени.

Возвращаемые значения:

- результат проверки штампа времени для заданной ЭП.

`TspVerifyResult TspStrVerifyCmsFile(TspVerifyParameters Param, string Icms)` - (соответствует функции *VCERT_TspStrVerifyCmsFile*)

Метод потоковой проверки штампа времени файла CMS

Аргументы:

- **Param** параметры проверки штампа времени CMS-сообщений;
- **Icms** файл (не нулевой длины) с подписанным CMS-сообщением, включающим штамп времени.

Возвращаемые значения:

- результат проверки штампа времени для заданной ЭП.

1.7.20 Получение online-статуса сертификата

Класс **OcspRequestParameters** - (соответствует структуре *ocsp_request_param_t*)

Параметры получения online-статуса сертификата:

- конструктор **OcspRequestParameters()**;
- свойство **OcspRequestFlags** **Flags**

Флаги (зарезервировано, должно быть равно **OcspRequestFlags.NoFlag**).

Класс **OcspResponseParameters** - (соответствует структуре *ocsp_response_param_t*)

Параметры вычисления ЭП online-статуса сертификата:

- конструктор **OcspResponseParameters()**;
- свойство **OcspResponseFlags** **Flags**

Флаги (зарезервировано, должно быть равно **OcspResponseFlags.NoFlag**).

Класс **OcspVerifyParameters** - (соответствует структуре *ocsp_verify_param_t*)

Параметры проверки ЭП online-статуса сертификата:

- конструктор **OcspVerifyParameters()**;
- свойство **OcspVerifyFlags** **Flags**

Флаги (зарезервировано, должно быть равно **OcspVerifyFlags.NoFlag**).

Класс **OcspVerifyResult** - (соответствует структуре *ocsp_verify_result_t*)

Результат проверки ЭП online-статуса сертификата:

- свойство **uint** **status**

Online-статус сертификата.

- свойство **uint** **reason**

Для аннулированного сертификата, причина аннулирования сертификата.

- свойство **DateTime** **revTime**

Для аннулированного сертификата, время аннулирования сертификата.

- свойство **DateTime** **thisUpdate**

Время начала действия данного online-статуса сертификата.

- свойство **DateTime** **nextUpdate**

Время окончания действия данного online-статуса сертификата.

- свойство **Certificate** **cert**

сертификат, на котором выполнялась проверка ЭП, т.е. сертификат сервера OSCP ответчика (равно **null** в случае ненахождения сертификата сервера OSCP ответчика).

Методы получения online-статуса сертификата

byte[] **OcspCreateRequest**(OcspRequestParameters Param, byte [] Cert) - (*соответствует функции VCERT_OcspCreateRequest*)

Метод создания запроса на получение online-статуса сертификата

Аргументы:

- **Param** параметры получения online-статуса сертификата;
- **Cert** блок памяти с сертификатом для получения online-статуса в DER-кодировке или PEM-формате.

Возвращаемые значения:

- блок памяти с созданным запросом на получение online-статуса сертификата.

byte[] **OcspSignResponse**(OcspResponseParameters Param, byte[] Request) - (*соответствует функции VCERT_OcspSignResponse*)

Метод обработки запроса на получение online-статуса сертификата

Аргументы:

- **Param** параметры вычисления ЭП online-статуса сертификата;
- **Request** блок памяти (не нулевой длины) с запросом на получение online-статуса сертификата.

Возвращаемые значения:

- блок памяти с подписанным online-статусом сертификата.

byte[] **OcspUrlObtainResponse**(OcspRequestParameters Param, string url, byte[] Cert) - (*соответствует функции VCERT_OcspUrlObtainResponse*)

Метод получения online-статуса сертификата на заданном OCSP сервере

Аргументы:

- **Param** параметры получения online-статуса сертификата;
- **url** строка (URI) с адресом сервера OCSP ответчика;
- **Cert** блок памяти с сертификатом для получения online-статуса в DER-кодировке или PEM-формате.

Возвращаемые значения:

- блок памяти с подписанным online-статусом сертификата.

OcspVerifyResult **OcspVerifyResponse**(OcspVerifyParameters Param, byte[] Response) - (*соответствует функции VCERT_OcspVerifyResponse*)

Метод проверки ЭП online-статуса сертификата

Аргументы:

- **Param** параметры проверки ЭП online-статуса сертификата;
- **Response** блок памяти (не нулевой длины) с подписанным online-статусом сертификата.

Возвращаемые значения:

- результат проверки ЭП online-статуса сертификата.

1.7.21 Протокол безопасности транспортного уровня TLS 1.2

Перечисление **TlsFlags**

Флаги создания контекста защищенного канала по протоколу TLS:

- **CreateClientSession**
соответствует флагу FLAG_TLS_CREAT_CLIENT_SESSION.
- **NoPeerVerification**
соответствует флагу FLAG_TLS_NO_PEER_VERIFICATION.
- **AcceptClientCertificate**
соответствует флагу FLAG_TLS_ACCEPT_CLIENT_CERTIF.
- **RequireClientCertificate**
соответствует флагу FLAG_TLS_REQUIRE_CLIENT_CERTIF.
- **NoRenegotiateResumption**
соответствует флагу FLAG_TLS_NO_RENEGO_RESUMPTION.
- **EnableUnsafeRenegotiation**
соответствует флагу FLAG_TLS_ENABLE_UNSAFE_RENEGO.
- **ProtocolVersion1_2**
соответствует флагу FLAG_TLS_PROTOCOL_VERSION_1_2.

TlsSessionCtx **TlsSessionCreate**(string Server, TlsFlags Flags) - (*соответствует функции VCERT_TlsSessionCreate*)

Метод создания контекста защищенного канала по протоколу TLS

Аргументы:

- **Server** строка с DNS-именем сервера (может использоваться только при создании контекста клиента). При значении, не равном **null**, производится верификация наличия указанного DNS-имени в дополнении "Альтернативное имя владельца" сертификата сервера;
- **Flags** флаги создания контекста защищенного канала по протоколу TLS.

Возвращаемые значения:

- класс контекста защищенного канала по протоколу TLS.

Класс TlsSessionCtx

Класс **TlsSessionCtx** содержит контекст библиотеки, на котором он был создан, поэтому объект класса **TlsSessionCtx** нельзя использовать, если объект класса **VcertObject**, при помощи которого он был создан, разрушен.

void TlsSessionDestroy() - (*соответствует функции VCERT_TlsSessionDestroy, вызывается в деструкторе класса TlsSessionCtx*)

Метод освобождения контекста защищенного канала по протоколу TLS

byte[] Handshake(byte[] Itls) - (*соответствует функции VCERT_*

TlsSessionHandshake)

Метод выполнения переговоров при создании защищенного канала TLS

Аргументы:

– **Itls** блок памяти с данными протокола TLS, полученными от противоположной стороны. При формировании первого сообщения клиентом длина должна быть установлена в **0**;

Возвращаемые значения:

– блок памяти с данными протокола TLS, отправляемыми противоположной стороне.

bool **Complete()** - *(соответствует функции VCERT_TlsSessionComplete)*

Метод определения состояния защищенного канала по протоколу TLS

Возвращаемые значения:

– **true** защищенный канал между клиентом и сервером уже установлен;
– **false** защищенный канал между клиентом и сервером еще не установлен и переговоры должны быть продолжены:

• при создании защищенного канала необходимо получить дополнительные данные от противоположной стороны и вызвать метод **Handshake()**;

• при переговорах, проходящих уже после создания защищенного канала, необходимо вызвать метод **Write()**, установив длину блока памяти с защищаемыми данными в **0**, и передать полученные данные TLS протокола противоположной стороне;

– генерирует **VcertException** в случае ошибки.

Перечисление **TlsQueryType**

Запрашиваемые данные или команда управления защищенным каналом TLS

– PeerCertificateBlob

соответствует флагу VCERT_TLS_QRY_PEER_CERTIF_BLOB.

– ProtocolVersion

соответствует флагу VCERT_TLS_QRY_PROTOCOL_VERSION.

– CipherSuiteValue

соответствует флагу VCERT_TLS_QRY_CIPH_SUITE_VALUE.

– CipherSuiteDescription

соответствует флагу VCERT_TLS_QRY_CIPH_SUITE_DESCR.

– SessionShutdown

соответствует флагу VCERT_TLS_QRY_SESSION_SHUTDOWN.

byte[] **Query**(TlsQueryType Type) - *(соответствует функции VCERT_TlsSessionQuery)*

Метод получения данных или управления защищенным каналом TLS

Аргументы:

– **Type** запрашиваемые данные или команда управления.

Возвращаемые значения:

– блок памяти с запрошенными данными или результатом команды управления.

byte[] **Write**(byte[] Itls) - (соответствует функции *VCERT_TlsSessionWrite*)
Метод отправки данных другой стороне защищенного канала TLS

Аргументы:

– **Itls** блок памяти (не нулевой длины) с защищаемыми данными, предназначенными для передачи противоположной стороне.

Возвращаемые значения:

– блок памяти с данными протокола TLS, отправляемыми противоположной стороне.

byte[] **Read**(byte[] Itls) - (соответствует функции *VCERT_TlsSessionRead*)
Метод получения данных от другой стороны защищенного канала TLS

Аргументы:

– **Itls** блок памяти (не нулевой длины) с данными протокола TLS, полученными от противоположной стороны.

Возвращаемые значения:

– блок памяти с защищаемыми данными, переданными противоположной стороной;

– генерируется исключение **VcertException** с кодом ошибки **VCERT_E_TLS_READ_MORE** - необходимо продолжить чтение данных от противоположной стороны. Перед чтением следует вызовом функции **Complete()** определить состояние защищенного канала и, при получении **false**, продолжить переговоры.

1.7.22 Преобразование в формат и из формата Base64

byte[] **ToBase64**(byte[] InBuf) - (соответствует функции *VCERT_EncodeMem*)

Метод закодирования бинарных данных в формат Base64

Аргументы:

– **InBuf** исходный буфер с бинарными данными для закодирования.

Возвращаемые значения:

– блок памяти с закодированными данными.

byte[] **FromBase64**(byte[] InBuf) - (соответствует функции *VCERT_DecodeMem*)

Метод раскодирования бинарных данных из формата Base64

Аргументы:

– **InBuf** исходный буфер с закодированными данными в кодировке Base64.

Возвращаемые значения:

- блок памяти с раскодированными данными.

1.7.23 Выработка случайного числа заданной длины

При вызове функции выработки случайного числа выполняется инициализация ДСЧ, если он не был инициализирован ранее.

void **GetRandom**(byte[] Random) - (соответствует функции VCERT_GenRandom)

Метод выработки случайного числа заданной длины

Аргументы:

- **Random** блок памяти для записи случайного числа.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД	База данных
ДСЧ	Датчик случайных чисел
КЗИ	Криптографическая защита информации
КС	Криптографический сервер
ЛСП	Локальный справочник пользователя (Local Certificate Store)
ОС	Операционная система (Operating System)
ПК	Программный комплекс
ПО	Программное обеспечение
ППО	Прикладное ПО
ПСП	Персональный справочник пользователя (Personal Security Environment)
САС	Список аннулированных сертификатов (Certificate Revocation List)
СЗИ	Средство защиты информации
СКЗИ	Система криптографической защиты информации
ССС	Сетевой справочник сертификатов (Network Certificate Store)
СУС	Система управления сертификатами (Public Key Infrastructure)
ФКН	Функциональный ключевой носитель
ЦР	Центр регистрации (Registration Authority)
ЦС	Центр сертификации (Certification Authority)
ЭД	Электронный документ
ЭП	Электронная подпись (Digital Signature)

ПЕРЕЧЕНЬ РИСУНКОВ

ПЕРЕЧЕНЬ ТАБЛИЦ

[illegible][illegible]