

ЦЕНТРАЛЬНЫЙ БАНК РОССИЙСКОЙ ФЕДЕРАЦИИ
(БАНК РОССИИ)

УТВЕРЖДЕН
ВАМБ.00115-06-ЛУ

**ПРИКЛАДНОЙ ПРОГРАММНЫЙ ИНТЕРФЕЙС
СРЕДСТВ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ ВЕРСИЯ 6**

**ПРИКЛАДНОЙ ПРОГРАММНЫЙ ИНТЕРФЕЙС
СКАД «СИГНАТУРА» ВЕРСИЯ 6 ДЛЯ ПЛАТФОРМЫ JAVA**

Руководство программиста

ВАМБ.00115-06 33 01

2020

Аннотация

Данный документ содержит описание библиотеки прикладного программного интерфейса (ППИ) программного комплекса (ПК) ВАМБ.00104-06 «Система криптографической авторизации электронных документов «Сигнатура» версия 6» (далее по тексту - СКАД «Сигнатура») для платформы Java, а также рекомендации по встраиванию и использованию библиотеки ППИ в прикладном программном обеспечении (ППО).

Документ предназначен для разработчиков прикладного программного обеспечения как руководство по встраиванию и использованию ППИ для платформы Java при работе со СКАД «Сигнатура» (далее - ППИ Java СКАД «Сигнатура»).

При встраивании библиотеки предполагается, что системный программист имеет знания о существующей архитектуре системы сертификатов открытых ключей, используемых рекомендациях и стандартах.

Документ разработан специалистами ООО «Валидата».

Содержание

1 БИБЛИОТЕКА ПРИКЛАДНОГО ПРОГРАММНОГО ИНТЕРФЕЙСА СКАД «СИГНАТУРА» ВЕРСИЯ 6 ДЛЯ ПЛАТФОРМЫ JAVA	6
1.1 Назначение библиотеки	6
1.2 Характеристики библиотеки	7
1.3 Использование библиотеки	9
1.3.1 Основные понятия и определения	9
1.3.2 Условия использования библиотеки	10
1.3.3 Описание состава библиотеки	11
1.4 Основной Java класс	11
1.5 Описание базовых типов	11
1.6 Описание структур блока памяти	12
1.7 Описание структур идентификации сертификата	12
1.8 Описание структур сертификата	13
1.9 Описание структур САС	16
1.9.1 Структуры инициализации	17
1.9.2 Функции инициализации	18
1.9.3 Функции деинициализации	20
1.10 Получение описаний ошибок	21
1.10.1 Функции получения описаний ошибок	21
1.11 Управление и вызов Мастеров	21
1.11.1 Функции управления и вызова Мастеров	21
1.12 Справочники и профили пользователя	24
1.12.1 Функции справочников и профилей пользователя	24
1.13 Визуализация объектов СУС	26
1.13.1 Функции визуализации объектов СУС	26
1.14 Экспорт и импорт объектов СУС	27
1.14.1 Структуры экспорта и импорта объектов СУС	27
1.14.2 Функции экспорта и импорта объектов СУС	28
1.14.3 Структуры разбора и получения информации об объектах СУС	29
1.14.4 Функции разбора и получения информации об объектах СУС	30
1.15 Построение и проверка цепочек объектов СУС	31
1.15.1 Структуры построения и проверки цепочек объектов СУС	31
1.15.2 Функции построения и проверки цепочек объектов СУС	32
1.16 Вычисление хэш-значений	33
1.16.1 Функции вычисления хэш-значений	33
1.17 Вычисление и проверка ЭП хэш-значений	35
1.17.1 Функции вычисления и проверки ЭП хэш-значений	35
1.18 Поиск и перечисление объектов СУС	35
1.18.1 Структуры поиска сертификатов	35
1.18.2 Функции поиска сертификатов	36
1.18.3 Функции перечисления объектов СУС	36
1.19 Вычисление ЭП CMS-сообщений	37
1.19.1 Структуры вычисления ЭП CMS-сообщений	37
1.19.2 Функции блочного вычисления совмещенной ЭП CMS-сообщений	38

1.19.3	Функции потокового вычисления совмещенной ЭП CMS-сообщений	38
1.19.4	Функции блочного вычисления отдельной ЭП CMS-сообщений	39
1.19.5	Функции потокового вычисления отдельной ЭП CMS-сообщений	40
1.20	Проверка ЭП CMS-сообщений	41
1.20.1	Структуры проверки ЭП CMS-сообщений	41
1.20.2	Функции блочной проверки совмещенных ЭП CMS-сообщений	43
1.20.3	Функции потоковой проверки совмещенных ЭП CMS-сообщений	44
1.20.4	Функции блочной проверки отдельных ЭП CMS-сообщений .	45
1.20.5	Функции потоковой проверки отдельных ЭП CMS-сообщений	46
1.21	Зашифрование CMS-сообщений	48
1.21.1	Структуры зашифрования CMS-сообщений	48
1.21.2	Функции блочного зашифрования CMS-сообщений	48
1.21.3	Функции потокового зашифрования CMS-сообщений	49
1.22	Расшифрование CMS-сообщений	50
1.22.1	Структуры расшифрования CMS-сообщений	50
1.22.2	Функции блочного расшифрования CMS-сообщений	51
1.22.3	Функции потокового расшифрования CMS-сообщений	51
1.23	Преобразование совмещенных и отдельных ЭП	53
1.23.1	Функции блочного преобразования отдельных ЭП в совмещенные	53
1.23.2	Функции блочного преобразования совмещенных ЭП в отдельные	53
1.23.3	Функции потокового преобразования совмещенных ЭП в отдельные	54
1.24	Получение информации о CMS-сообщениях	55
1.24.1	Структуры получения информации о CMS-сообщениях	55
1.24.2	Функции блочного получения информации о CMS-сообщениях	59
1.24.3	Функции потокового получения информации о CMS-сообщениях	59
1.25	Простановка и проверка штампов времени	60
1.25.1	Структуры простановки и проверки штампов времени	60
1.25.2	Функции блочной простановки и проверки штампов времени .	62
1.25.3	Функции потоковой простановки и проверки штампов времени	64
1.26	Получение online-статуса сертификата	67
1.26.1	Структуры получения online-статуса сертификата	67
1.26.2	Функции получения online-статуса сертификата	68
1.27	Протокол безопасности транспортного уровня TLS 1.2	70
1.27.1	Функции протокола безопасности транспортного уровня TLS 1.2	70
1.28	Преобразование в формат и из формата Base64	72
1.28.1	Функции преобразования в формат и из формата Base64	72
1.29	Выработка случайного числа заданной длины	73
1.29.1	Функции выработки случайного числа заданной длины	73
1.30	Описание конфигурационного файла pk1.conf, XML-шаблона и XML-запроса	73

2 ОПИСАНИЕ ОШИБОЧНЫХ СИТУАЦИЙ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ	80
ПЕРЕЧЕНЬ ТАБЛИЦ	82

1 БИБЛИОТЕКА ПРИКЛАДНОГО ПРОГРАММНОГО ИНТЕРФЕЙСА СКАД «СИГНАТУРА» ВЕРСИЯ 6 ДЛЯ ПЛАТФОРМЫ JAVA

1.1 Назначение библиотеки

ВАМБ.00115-06 «Библиотека прикладного программного интерфейса (ППИ) СКАД "Сигнатура" версия 6 для платформы Java» (далее по тексту - библиотека или ППИ Java) обеспечивает функции аутентификации в соответствии с рекомендациями Х.509, электронной подписи (ЭП) по ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012 и шифрования по ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015, ГОСТ 28147-89 в среде выполнения Java JRE версия 1.6, 1.7 или 1.8. Реализация криптографических функций, доступных посредством библиотеки, основана на использовании программного обеспечения (ПО) СКАД "Сигнатура".

Примечания

1 Для обеспечения возможности плавного перехода с ППИ Java СКАД "Сигнатура" версия 5 на ППИ Java СКАД "Сигнатура" версия 6 функционал ППИ Java версии 6 включает в себя функционал библиотеки из состава ППИ Java версии 5. В связи с этим прикладные системы, в которые была встроена ППИ Java версии 5, будут корректно функционировать совместно с ППИ Java версии 6.

2 Работы по оценке влияния для прикладных систем, функционирующих совместно с ППИ Java СКАД "Сигнатура" версия 6, должны проводиться только после доработки указанных прикладных систем в соответствии с настоящим руководством.

Библиотека обеспечивает обращение к следующим функциям:

- генерация ключа ЭП и/или закрытого ключа шифрования по ГОСТ Р 34.10-2012 (для ключей ЭП и закрытых ключей шифрования длиной 256 и 512 бит);
- формирование запроса на получение сертификата ключа проверки ЭП и/или открытого ключа шифрования в формате PKCS#10;
- формирование запроса на аннулирование сертификата ключа проверки ЭП и/или открытого ключа шифрования;
- выработка хэш-значения для файла и области памяти по ГОСТ Р 34.11-94 и ГОСТ Р 34.11-2012 (для хэш-значений длиной 256 и 512 бит);
- вычисление ЭП файла и области памяти по ГОСТ Р 34.10-2012 (для ключей ЭП длиной 256 и 512 бит);
- проверка ЭП файла и области памяти по ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012 (для ключей проверки ЭП длиной 512 и 1024 бита);
- удаление ЭП из файла и области памяти, преобразование совмещенных (Attached) и отделенных (Detached) ЭП;
- вычисление ЭП хэш-функции данных по ГОСТ Р 34.10-2012 (для ключей ЭП длиной 256 и 512 бит);

- проверка ЭП хэш-функции данных по ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012 (для ключей проверки ЭП длиной 512 и 1024 бита);
- зашифрование и расшифрование файла и области памяти по ГОСТ 28147-89 и ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015 (блочные шифры «Магма» и «Кузнечик»);
- реализация механизма простановки и проверки штампов времени ЭП в соответствии с RFC 3161;
- реализация протокола безопасности транспортного уровня TLS 1.2;
- преобразование бинарных данных в формат и из формата Base64;
- выработка случайного числа заданной длины.

Примечание — Устаревшие ГОСТ Р 34.11-94 и ГОСТ Р 34.10-2001 должны использоваться исключительно для проверки ЭП документов из доверенных архивов.

Форматы сертификатов ключей проверки ЭП и/или открытых ключей шифрования, списков аннулированных сертификатов (САС) и PKCS#10 запросов на получение сертификатов, формируемых и поддерживаемых библиотекой, соответствуют Рекомендациям по стандартизации Р 1323565.1.023-2022 «Использование алгоритмов ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012 в сертификате, списке аннулированных сертификатов (CRL) и запросе на сертификат PKCS#10 инфраструктуры открытых ключей X.509».

Форматы защищенных сообщений (CMS-сообщений), формируемых и поддерживаемых библиотекой — файлов и областей памяти, подписанных по ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012 или зашифрованных по ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015 (блочные шифры «Магма» и «Кузнечик») — соответствуют Рекомендациям по стандартизации Р 1323565.1.025-2019 «Использование алгоритмов ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015, ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012 в сообщениях формата Cryptographic Message Syntax (CMS)».

Реализация протокола безопасности транспортного уровня TLS 1.2 библиотеки соответствует Рекомендациям по стандартизации Р 1323565.1.020-2018 «Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.2)».

Для лучшего понимания данного документа рекомендуется ознакомиться с упрощенным описанием стандарта **ASN.1 A Layman's Guide to a Subset of ASN.1, BER, and DER**, опубликованным компанией **RSA Laboratories** (см. статью на английском языке по ссылке <http://luca.ntop.org/Teaching/Appunti/asn1.html>).

1.2 Характеристики библиотеки

Библиотека ППИ для платформы Java предназначена для встраивания СКАД "Сигнатура" в прикладные системы. ППИ Java СКАД "Сигнатура" предоставляет программам, написанным на языке Java, доступ к набору констант, классов и функций библиотеки упрощенного ППИ для работы с сертификатами СКАД "Сигнатура". ППИ Java СКАД "Сигнатура" разработан в соответствии со спецификацией Oracle Java™ Native Interface 6.0 API Specification (в дальнейшем JNI), входящей в состав Oracle Java™ Platform, Standard Edition 6 (в дальнейшем

Java™ SE).

Требования к аппаратно-программной среде, в которой функционирует библиотека, приведены в документе ВАМБ.00107-06 30 01 «СКАД «Сигнатура» версия 6. АПК «Средство КЗИ СКАД «Сигнатура» версия 6». Формуляр».

Состав и функциональность процедур, форматы обращений из прикладного программного обеспечения (ПО) к функциям ППИ Java СКАД "Сигнатура" соответствуют составу и функциональности процедур, форматам обращений библиотеки упрощенного ППИ для работы с сертификатами СКАД "Сигнатура", соответственно, за исключением различий, обусловленных особенностями языка программирования Java.

Функции библиотеки, вычисляющие хэш-значения сообщений и оперирующие CMS-сообщениями, в большинстве случаев имеют две реализации — блочную и потоковую. Блочные функции обрабатывают сообщение целиком (в один блок), потоковые функции позволяют обрабатывают сообщения небольшими порциями (потоком).

При использовании блочных функций в 32-битном прикладном программном обеспечении (ПО) обеспечивается выполнение функций под файлом или областью памяти объемом максимум до 500 Мбайт, а в 64-битном прикладном ПО — объемом максимум до 2 Гбайт. При этом данные величины могут быть уменьшены в зависимости от текущего использования и гранулированности виртуальной памяти вызывающего процесса прикладного ПО.

При использовании потоковых функций как в 32-битном прикладном ПО, так и в 64-битном прикладном ПО обеспечивается выполнение функций под файлом или областью памяти без ограничения общего объема.

В выполнении каждой потоковой операции с областью памяти задействованы по три потоковые функции — инициализации, продолжения и финализации. Функция инициализации вызывается один раз и, при ее успешном завершении, возвращает контекст начатой потоковой операции. Функция продолжения может вызываться один или несколько раз — для каждого обрабатываемого блока памяти; при ошибке выполнения данной функции контекст потоковой операции освобождается и более использован быть не может. Функция финализации вызывается один раз, в конце выполнения потоковой операции; после завершения данной функции контекст потоковой операции освобождается и более использован быть не может.

При выборе блочных или потоковых функций следует учитывать, что:

- блочные функции требуют объем памяти от двух до четырех объемов исходного файла или области памяти (т.е. блочные функции потребляют больше памяти), при этом потоковые функции выполняются несколько медленнее, чем блочные функции (т.е. потоковые функции менее производительные);

- форматы защищенных (подписанных и зашифрованных) данных, обрабатываемые блочными и потоковыми функциями, идентичны. Таким образом, защищенные данные, сформированные блочной функцией, в большинстве случаев могут быть обработаны потоковыми функциями, и наоборот - с уче-

том ограничений максимального обрабатываемого объема данных. Ниже в документе описаны специальные случаи, когда для обработки конкретного типа (блочного или потокового) защищенных данных необходимо использовать соответствующие этому типу (блочные или потоковые) функции.

Примечания

1 В данном документе термин **закрытый ключ** используется для обозначения ключа ЭП и/или закрытого ключа шифрования, а термин **открытый ключ** — для обозначения ключа проверки ЭП и/или открытого ключа шифрования.

2 В данном документе термин **URI** (сокращение от Uniform Resource Identifier) используется для обозначения строки, идентифицирующей тип и местонахождение справочников сертификатов, а также местонахождение точек доступа к центру (AIA) и точек распространения CAC (CDP).

3 Исполняемый модуль **spki1jni.dll** библиотеки всегда имеет такую же битность, как и ПО ППИ Java СК АД "Сигнатура", в состав которого он входит. Битность исполняемого модуля **spki1jni.dll** библиотеки должна совпадать с битностью использующего его прикладного ПО.

1.3 Использование библиотеки

1.3.1 Основные понятия и определения

В данном документе под профилем пользователя подразумевается логическое объединение следующих сущностей:

- **закрытый ключ** — загрузка закрытого ключа необходима для возможности вычисления ЭП и расшифрования. При загрузке закрытого ключа выполняется инициализация датчика случайных чисел (ДСЧ), если он не был проинициализирован ранее;

- **персональный справочник пользователя (ПСП)** — защищенное хранилище, содержащее сертификаты доверенных (корневых) Центров сертификации (ЦС). ПСП защищено ЭП, вычисленной при его формировании на закрытом ключе профиля пользователя. Проверка ЭП ПСП выполняется на сертификате пользователя - называемом далее **рабочим сертификатом**, открытый ключ которого является парным для закрытого ключа профиля пользователя. Для хранения ПСП может использоваться файл — подписанное CMS-сообщение, а также системное хранилище сертификатов ОС Microsoft Windows;

- **локальный справочник пользователя (ЛСП)** — незащищенное хранилище, содержащее сертификаты ЦС второго и более низких уровней, списки аннулированных сертификатов (САС) ЦС любых уровней, сертификаты Центров регистрации (ЦР), рабочий сертификат, запросы PKCS#10, запросы на аннулирование, а также сторонние сертификаты. Для хранения ЛСП могут использоваться базы данных (БД) GDBM и ODBC, а также системное хранилище сертификатов ОС Microsoft Windows;

- **сетевой справочник сертификатов (ССС)** — незащищенное хранилище (опциональное), содержащее сертификаты ЦС второго и более низких уровней, САС ЦС любых уровней, сертификаты ЦР, а также сторонние сертификаты. Доступ к СССР осуществляется по стандартному протоколу **Lightweight Directory**

Access Protocol (LDAP).

Все операции, выполняемые посредством вызовов функций библиотеки, оперируют объектами **системы управления сертификатами (СУС)** — сертификатами, САС, запросами PKCS#10, запросами на аннулирование, содержащимися в хранилищах ПСП, ЛСП и (опционально) ССС. Единственным исключением из этого правила является возможность загрузки сертификатов ЦС второго и более низких уровней из точек AIA и САС всех уровней из точек CDP при построении и проверке цепочек сертификации объектов СУС.

Криптографические операции вычисления ЭП и расшифрования используют загруженный в процессе инициализации контекста библиотеки закрытый ключ. Поиск сертификатов ЦС или ЦР для проверки ЭП обновления - защищенного сообщения, содержащего объекты СУС для последующей загрузки в ПСП и ЛСП — производится в ПСП, ЛСП и (опционально) ССС. Поиск сертификатов подписантов при проверке ЭП подписанных CMS-сообщений, а также поиск сертификатов получателей при зашифровании CMS-сообщений производится в ЛСП и (опционально) ССС. Формируемые запросы PKCS#10 и запросы на аннулирование сохраняются в ЛСП.

Для решения задач, не требующих личного закрытого ключа и сертификата, в библиотеке реализована поддержка так называемых **анонимных** профилей. В **анонимном** профиле в качестве ПСП фигурирует ПСП доверенного (корневого) ЦС. Во время инициализации контекста библиотеки для проверки ЭП такого ПСП требуется один из сертификатов данного доверенного (корневого) ЦС, необходимо содержащийся в самом ПСП.

Анонимные профили могут быть использованы при проверке ЭП подписанных CMS-сообщений. Также **анонимным** может быть профиль клиентской стороны защищенного канала, установленного по протоколу TLS в том случае, когда серверная сторона не выполняет аутентификацию клиентской стороны (т.е. когда задействована односторонняя аутентификация).

1.3.2 Условия использования библиотеки

При использовании библиотеки необходимо соблюдать следующие условия:

- библиотека предназначена для использования в приложениях, написанных на языке программирования Java;
- при использовании библиотеки в приложениях, написанных на других языках программирования, должно выполняться требуемое преобразование форматов данных и параметров вызова функций.
- если не указано обратное, все строковые данные, получаемые и возвращаемые библиотекой, должны быть в кодировке Windows Code Page 1251 (Windows-1251, CP1251). В частности, структуры объектов СУС всегда должны содержать данные в указанной кодировке.

Приложение должно вызывать функции библиотеки последовательно:

- начало работы приложения;
- инициализация минимального контекста библиотеки функцией **VCERT_**

InitMinimal(). Данная операция необходима для корректной работы с различными (в том числе локализованными) ресурсами библиотеки;

- инициализация полноценного контекста библиотеки функцией **VCERT_Initialize()** или **VCERT_InitializeEx()**;

- использование полноценного контекста - вызов функций библиотеки для вычисления и проверки ЭП, зашифрования и расшифрования, поиска сертификатов, и т.п.;

- освобождение полноценного контекста библиотеки функцией **VCERT_Uninitialize()**;

- освобождение минимального контекста библиотеки функцией **VCERT_Uninitialize()**;

- окончание работы приложения.

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

1.3.3 Описание состава библиотеки

В состав библиотеки входят следующие файлы:

- spki1jni.dll - модуль динамической библиотеки JNI интерфейса;
- Pki1.LocalIface.Jar - набор Java классов библиотеки интерфейса;
- Pki1.Test.Jar - набор Java классов тестовой утилиты;
- Pki1.Test.Cmd - командный файл для запуска тестовой утилиты.

1.4 Основной Java класс

Все нижеописанные константы, методы и классы принадлежат основному Java классу библиотеки ППИ Pki1.LocalIface. При вызове одной из функций инициализации (VCERT_Initialize(), VCERT_InitializeEx() или VCERT_InitMinimal()) создается контекст библиотеки упрощенного ППИ для работы с сертификатами СКАД "Сигнатура" версия 6, который сохраняется в объекте основного класса интерфейса, а при вызове функции деинициализации (VCERT_Uninitialize()) данный контекст освобождается.

Константы, методы и классы Pki1.LocalIface соответствуют объектам библиотеки прикладного программного интерфейса СКАД "Сигнатура". Рекомендуется ознакомиться с подробным описанием соответствующих объектов в документе ВАМБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

1.5 Описание базовых типов

```
public static class strhash_handle_t
```

Контекст потокового хэширования данных.

```
public static class strsign_handle_t
```

Контекст потокового вычисления ЭП данных.

public static class **strverify_handle_t**

Контекст потоковой проверки ЭП данных.

public static class **strencrypt_handle_t**

Контекст потокового зашифрования данных.

public static class **strdecrypt_handle_t**

Контекст потокового расшифрования данных.

public static class **enuobj_handle_t**

Контекст перебора объектов справочника (персонального, локального, сетевого).

public static class **strcms_handle_t**

Контекст выполнения потоковых функций CMS-сообщений.

public static class **tls_handle_t**

Контекст защищенного канала (сеанса связи) по протоколу TLS.

1.6 Описание структур блока памяти

Структура mem_blk_t

Непрерывный блок памяти (блок данных):

– int **len**

Длина блока памяти в байтах, должна быть в интервале от 0 до $2^{31} - 1$.

– byte[] **buf**

Блок памяти, может быть равен NULL только если длина блока равна 0.

1.7 Описание структур идентификации сертификата

Структура issuer_and_serial_t

Пара издателя и серийного номера сертификата:

– String **issuer**

Строка с X.500-именем издателя сертификата.

– String **serialNumber**

Строка с серийным номером сертификата.

Структура certid_t

Уникальный идентификатор сертификата:

– int **type**

Выбор способа идентификации: ID_ISSUER_AND_SERIAL, ID_KEYID или ID_CERTHASH.

– issuer_and_serial_t **ias**

Структура пары издателя и серийного номера сертификата - должна быть заполнена при идентификации по ID_ISSUER_AND_SERIAL.

- String **keyId**

Строка идентификатора закрытого ключа, соответствующего данному сертификату - должна быть заполнена при идентификации по ID_KEYID.

- mem_blk_t **certHash**

Блок памяти с хэш-значением пары имени издателя и серийного номера сертификата - должен быть заполнен при идентификации по ID_CERTHASH.

1.8 Описание структур сертификата

Структура **altname_t**

Альтернативное имя издателя или владельца сертификата:

- String **emailAddress**

Строка с адресом электронной почты RFC 822.

- String **DNS**

Строка с именем системы имен доменов DNS.

- String **URI**

Строка с уникальным адресом ресурса URI.

- String **IP**

Строка с адресом IP протокола.

- String **organizationName**

Строка с наименованием организации.

- String **registredAddress**

Строка с зарегистрированным адресом.

- String **surname**

Строка с Ф.И.О. владельца сертификата.

- String **businessCategory**

Строка с должностью владельца сертификата.

- String **telephoneNumber**

Строка с номером телефона владельца сертификата.

- String **description**

Строка с описанием в свободной форме.

- String **account_number**

Строка с номером расчетного счета.

- String **bank_id**

Строка с банковским идентификационным кодом (БИК).

- String **physicalDelivery**

Строка с почтовым адресом.

- String **exchange_address**

Строка с адресом Microsoft Exchange.

Структура **policy_t**

Регламент (политика) использования сертификата:

- String **oid**

Строка объектного идентификатора (OID), идентифицирующего регламент сертификата.

Структура extkeyusage_t

Расширенная область использования открытого ключа сертификата:

– String **oid**

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

Структура extension_t

Дополнение (расширение) X.509 сертификата:

– String **oid**

Строка объектного идентификатора (OID), идентифицирующего дополнение сертификата.

– int **type**

ASN.1 тип дополнения (4 - OctetString; 12 - UTF8String; 16 - Sequence; 22 - IA5String).

– int **critical**

Признак критичности дополнения.

– int **len**

Длина блока памяти дополнения в байтах.

– byte[] **data**

Блок памяти дополнения в DER-кодировке. Для дополнений, закодированных в виде ASN.1 строки, возвращается внутреннее содержимое строки в кодировке, соответствующей типу ASN.1 строки.

Структура certificate_t

Структура, описывающая сертификат:

– int **fields**

Содержит битовую маску (побитовое ИЛИ) тех и только тех констант FIELD_XXX, для которых в соответствующих им полях структуры заданы значения (например, при задании FIELD_SUBJECT | FIELD_CERTHASH заполнены только поля subject и certHash).

– String **issuer**

Поле с X.500-именем издателя сертификата в виде строки (при задании FIELD_ISSUER в поле fields).

– String **serialNumber**

Поле с серийным номером сертификата в виде строки (при задании FIELD_SERIAL в поле fields).

– String **subject**

Поле с X.500-именем владельца сертификата в виде строки (при задании FIELD_SUBJECT в поле fields). При заполнении шаблона сертификата для поиска или зашифрования имя владельца сертификата можно задавать частично и/или включать в него подстановочные знаки-маски "" и "?":*

- *при выполнении поиска сертификатов в кэше библиотеки или в ЛСП, расположенном в БД GDBM, допускается подавать строку, заполненную частично с подстановочными знаками-масками "*" и "?" (например,*

"*,OU=123456789,*"). В этом случае будет производиться перебор всех сертификатов на соответствие имени владельца сертификата указанному в шаблоне регулярному выражению;

- при выполнении поиска сертификатов в ЛСП, расположенном в БД ODBC, в кэше библиотеки или в CCC, допускается подавать строку, заполненную частично (например, "OU=7395399001"). В этом случае, при задании флага поиска FLAG_FIND_PARTIAL_SUBJECT или FLAG_ENCRYPT_PARTIAL_SUBJECT, все сертификаты, находящиеся в справочнике, будут проверяться на соответствие указанной в шаблоне части имени владельца;

- при выполнении поиска сертификатов в CCC допускается подавать строку, заполненную частично с подстановочным знаком-маской "*" (например, "*,INN=770200040698,*"). В этом случае, при задании флага поиска FLAG_FIND_SUBJECT_ATTRIBUTE или FLAG_ENCRYPT_SUBJECT_ATTRIBUTE, будет производиться поиск сертификатов во всех контейнерах директории, содержащих в значении атрибута LDAP vdSubjName указанное в шаблоне регулярное выражение.

– String **algorithm**

Поле со строкой алгоритма открытого ключа сертификата (при задании FIELD_ALGORITHM в поле fields).

– long **notBeforeEx**

Поле с датой начала действия сертификата (при задании FIELD_NOTBEFORE в поле fields).

– long **notAfterEx**

Поле с датой окончания действия сертификата (при задании FIELD_NOTAFTER в поле fields).

– int **keyUsage**

Поле с маской (побитовым ИЛИ) разрешенных областей использования открытого ключа (при задании FIELD_KEYUSAGE в поле fields).

– long **notBeforePrivateEx**

Поле с датой начала действия закрытого ключа сертификата (при задании FIELD_NOTBEFOREPRIVATE в поле fields).

– long **notAfterPrivateEx**

Поле с датой окончания действия закрытого ключа сертификата (при задании FIELD_NOTAFTERPRIVATE в поле fields).

– altname_t **issuerAltName**

Поле со структурой альтернативного имени издателя сертификата (при задании FIELD_ISSUERALTNAME в поле fields).

– altname_t **subjectAltName**

Поле со структурой альтернативного имени владельца сертификата (при задании FIELD_SUBJECTALTNAME в поле fields).

– String **keyId**

Поле со строковым идентификатором закрытого ключа, соответствующего данному сертификату (при задании FIELD_KEYID в поле fields). Значение данного поля уникально для каждого сертификата.

– int **policy_num**

Количество элементов массива регламентов использования сертификата (при задании FIELD_POLICY в поле fields).

– policy_t[] **policies**

Поле с массивом регламентов использования сертификата (при задании FIELD_POLICY в поле fields).

– int **extkeyusage_num**

Количество элементов массива расширенных областей использования открытого ключа (при задании FIELD_EXTKEYUSAGE в поле fields).

– extkeyusage_t[] **extKeyUsage**

Поле с массивом расширенных областей использования открытого ключа (при задании FIELD_EXTKEYUSAGE в поле fields).

– int **extension_num**

Количество элементов массива дополнений (расширений) сертификата (при задании FIELD_EXTENSIONS в поле fields).

– extension_t[] **extensions**

Поле с массивом дополнений (расширений) сертификата (при задании FIELD_EXTENSIONS в поле fields).

– mem_blk_t **certEncoded**

Поле с блоком памяти, содержащим сертификат в DER-кодировке (при задании FIELD_CERTENCODED в поле fields).

– mem_blk_t **certHash**

Поле с блоком памяти, содержащим хэш-значение пары имени издателя и серийного номера сертификата (при задании FIELD_CERTHASH в поле fields).

1.9 Описание структур САС

Структура revcert_t

Аннулированный сертификат:

– String **serialNumber**

Поле с серийным номером аннулированного сертификата в виде строки.

– long **revtimeEx**

Поле со временем аннулирования сертификата.

– int **reason**

Поле с причиной аннулирования сертификата:

- **-1** - причина отсутствует;
- **0** - причина не указана;
- **1** - компрометация ключа;
- **2** - компрометация ключа ЦС;
- **3** - изменена принадлежность;
- **4** - сертификат заменен;
- **5** - действие сертификата остановлено;
- **6** - действие сертификата приостановлено;

- **8** - удаление из САС (данное значение может присутствовать только в запросе на аннулирование сертификата);
- **9** - привилегия аннулирована;
- **10** - компрометация ключа ЦР.

Структура **crl_t**

Структура, описывающая САС:

– int **fields**

Содержит побитовое ИЛИ тех и только тех констант *FIELD_CRL_XXX*, для которых в соответствующих им полях структуры заданы значения (например, при задании *FIELD_CRL_ISSUER* заполнено только поле *issuer*).

– String **issuer**

Поле с X.500-именем издателя САС в виде строки (при задании *FIELD_CRL_ISSUER* в поле *fields*).

– long **lastUpdateEx**

Поле со временем начала действия САС (при задании *FIELD_CRL_LASTUPDATE* в поле *fields*).

– long **nextUpdateEx**

Поле со временем окончания действия САС (при задании *FIELD_CRL_NEXTUPDATE* в поле *fields*).

– int **number**

Поле с порядковым номером САС (при задании *FIELD_CRL_NUMBER* в поле *fields*).

– int **revcert_num**

Количество элементов массива аннулированных сертификатов (при задании *FIELD_CRL_REVOKED* в поле *fields*).

– revcert_t[] **revcerts**

Поле с массивом аннулированных сертификатов (при задании *FIELD_CRL_REVOKED* в поле *fields*).

– mem_blk_t **crlEncoded**

Поле с блоком памяти, содержащим САС в DER-кодировке (при задании *FIELD_CRL_CRL_ENCODED* в поле *fields*).

– mem_blk_t **crlHash**

Поле с блоком памяти, содержащимся в дополнении "Идентификатор ключа владельца" сертификата издателя САС (при задании *FIELD_CRL_CRL_HASH* в поле *fields*). Либо хэш-значение имени издателя САС, либо значение данного поля является первичным ключом при поиске САС в справочниках.

1.9.1 Структуры инициализации

Структура **local_param_t**

Параметры инициализации контекста локальной библиотеки:

– int **flag**

Поле с маской (побитовым ИЛИ) флагов инициализации контекста библиотеки (*FLAG_INIT_XXX*).

– String **pse**

Поле с путем (URI) к ПСП следующего вида:

- **pse://signed/C:\Users\TestUser\local.pse** - ПСП, расположенный в файловом хранилище, где **pse://signed/** - префикс, а **C:\Users\TestUser\local.pse** - имя файла ПСП;

- **system://pse/[lm/]00:01:02:03:0C:0D:0E:0F** - ПСП, расположенный в системном хранилище ОС Microsoft Windows, где **system://pse/** - префикс, **lm/** - опциональный признак использования хранилища компьютера (если не указан - используется хранилище пользователя), а **00:01:02:03:0C:0D:0E:0F** - данные дополнения "Идентификатор ключа владельца" рабочего сертификата.

– String **localstore**

Поле с путем (URI) к ЛСП следующего вида:

- **file://C:\Users\TestUser\local.gdbm** - ЛСП, расположенный в БД GDBM, где **file://** - префикс, а **C:\Users\TestUser\local.gdbm** - имя файла БД ЛСП.

- **odbc://TESTDSN** - ЛСП, расположенный в БД ODBC, где **odbc://** - префикс, а **TESTDSN** - имя источника данных (Data Source Name, DSN) БД ЛСП;

- **system://local/[lm/]** - ЛСП, расположенный в системном хранилище ОС Microsoft Windows, где **system://local** - префикс, а **lm/** - опциональный признак использования хранилища компьютера (если не указан - используется хранилище пользователя).

– String **ldap**

Поле с путем (URI) к CCC следующего вида:

- **ldap://[simple/|negotiate/][user[:pass]@]host[:389]/DC=RU** - CCC, доступный по протоколу LDAP, где **ldap://** - префикс, **host** - DNS-имя узла сервера, **389** - TCP-порт подключения (это значение используется по умолчанию), **DC=RU** - базовый контейнер CCC. Модификатор **simple/** указывает, что следует использовать простой механизм аутентификации с CCC, а модификатор **negotiate/** задает использование механизма аутентификации Kerberos. С помощью параметров **user** и **pass** можно задать имя пользователя и его пароль, которые будут использоваться при аутентификации во время подключения к CCC.

– String **pincode**

Поле с ПИН-кодом ключевого носителя типа смарт-карта в виде строки. Задание данного поля позволяет загружать закрытые ключи с носителей типа смарт-карта без выдачи пользовательского интерфейса ввода ПИН-кода.

1.9.2 Функции инициализации

Перед использованием любой из функций библиотеки (за исключением функций инициализации) библиотеку необходимо инициализировать - получить контекст библиотеки с помощью одной из функций инициализации.

Контекст библиотеки, возвращаемый любой из таких функций, является безопасным для параллельного (многопоточного) использования - т.е. данный контекст может быть использован в вызовах функций библиотеки параллельно из

нескольких потоков. При этом сами функции инициализации и деинициализации не являются безопасными для параллельного (многопоточного) использования:

- в рамках данного конкретного процесса операционной системы (ОС) все вызовы функций инициализации и деинициализации должны быть сериализованы, т.е. должны выполняться последовательно;
- в рамках различных процессов ОС, выполняющихся от имени данного конкретного пользователя, функции инициализации и деинициализации можно вызывать параллельно при условии, что они не модифицируют настройки профилей пользователя ПК "Справочник сертификатов" из Реестра.

При доступе к ЛСП, расположенному в БД GDBM, следует иметь в виду, что данный тип БД не обеспечивает возможность параллельного доступа - т.е. к конкретному ЛСП разрешено иметь доступ одновременно только из одного процесса, и только посредством одного контекста библиотеки. При необходимости осуществления параллельного доступа к конкретному ЛСП из нескольких процессов или посредством нескольких контекстов библиотеки следует пользоваться ЛСП, расположенным в БД ODBC.

int VCERT_InitMinimal ()

Функция инициализации минимального контекста библиотеки

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

Функция инициализации минимального контекста не осуществляет доступ к ПСП, ЛСП и/или ССС.

С минимальным контекстом допускается вызывать исключительно следующие функции библиотеки:

- вычисления хэш-значения и формирования случайной последовательности;
- выполнения преобразования совмещенной (Attached) и отделенной (Detached) ЭП;
- выполнения преобразования бинарных данных в/из формата Base64;
- получения информации о защищенных (подписанных и зашифрованных) данных;
- разбора и отображения интерфейса с данными объектов СУС;
- запуска Мастеров формирования запросов PKCS#10 и управления настройками профилей пользователя;
- клиентской стороны протокола безопасности транспортного уровня TLS 1.2;
- получения внутреннего стека ошибок низкоуровневой библиотеки.

int VCERT_Initialize (String profile, int flag)

Упрощенная функция инициализации контекста библиотеки

Аргументы:

– **profile** (in) строка с именем профиля из настроек конфигурационного файла **pki1.conf**, или из настроек профилей пользователя ПК "Справочник сертификатов" при указании флага инициализации **FLAG_INIT_CERTSTOREPROFILE**. Если в конфигурационном файле **pki1.conf** указан профиль по умолчанию, то данная строка может быть равна NULL. При использовании локальной библиотеки поддерживаются следующие дополнительные возможности:

- задание значения "MY" позволяет использовать настройки профилей пользователя ПК "Справочник сертификатов" и пользовательский интерфейс выбора профиля без задания флага инициализации **FLAG_INIT_CERTSTOREPROFILE**;

- задание флага инициализации **FLAG_INIT_CERTSTOREPROFILE** позволяет указать требуемый профиль пользователя ПК "Справочник сертификатов" по имени;

– **flag** (in) маска (побитовое ИЛИ) флагов инициализации (при использовании локальной библиотеки могут быть **FLAG_INIT_XXX**, иначе значение зарезервировано и не используется).

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_InitializeEx** (int prov, Object param)

Расширенная функция инициализации контекста библиотеки

Аргументы:

– **prov** (in) тип используемой библиотеки (**CRYPT_LOCAL** для локальной библиотеки);

– **param** (in) структура параметров инициализации контекста (структуру **local_param_t** для локальной библиотеки).

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.9.3 Функции деинициализации

int **VCERT_Uninitialize** ()

Функция деинициализации (освобождения) контекста библиотеки

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

Данная функция выполняет деинициализацию (освобождение) контекста библиотеки и связанных с ним ресурсов, выделенных в функциях **VCERT_InitMinimal()**, **VCERT_Initialize()** и **VCERT_InitializeEx()**, а также производит выгрузку загруженных для этого контекста закрытых ключей.

1.10 Получение описаний ошибок

1.10.1 Функции получения описаний ошибок

String **VCERT_GetErrorText** (int error)

Функция получения текстового описания ошибки по ее коду

Аргументы:

- **error** (in) код ошибки.

Возвращаемые значения:

- **буфер** с текстовым описанием ошибки, **NULL** - в случае ошибки.

int **VCERT_GetErrorLineEx** (mem_blk_t error, mem_blk_t file, int[] line, mem_blk_t data)

Функция получения стека внутренних ошибок библиотеки

Аргументы:

- **error** (out) блок памяти с кодом внутренней ошибки и ее текстовым описанием - строкой, заканчивающейся символом '\0';
- **file** (out) блок памяти с именем файла исходных текстов, в котором возникла внутренняя ошибка - строкой, заканчивающейся символом '\0';
- **line** (out) номер строки файла исходных текстов, в которой возникла внутренняя ошибка (параметр должен представлять собой массив с одним элементом);
- **data** (out) блок памяти с расширенными данными внутренней ошибки - строкой, заканчивающейся символом '\0'.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки. Для получения полного стека функцию следует вызывать в том же потоке, что и завершившуюся ошибкой исходную функцию до тех пор, пока возвращаемое ею значение равно **VCERT_OK**.

Для возможности корректного преобразования содержимого блоков памяти **error**, **file** и **data** в строковые данные необходимо удалить находящийся в конце этих блоков символ '\0'.

1.11 Управление и вызов Мастеров

1.11.1 Функции управления и вызова Мастеров

int **VCERT_ControlEx** (long hWnd, int cmd, mem_blk_t idat, mem_blk_t odat, int flag)

Функция управления контекстом и библиотекой, вызова Мастеров

Аргументы:

- **hWnd** (in) идентификатор родительского окна. В ОС Microsoft Windows в качестве идентификатора родительского окна используется значение, возвращаемое системной функцией **GetActiveWindow()**;

– **cmd** (in) код команды, которую необходимо выполнить (флаги VCERT_CMD_XXX):

- **VCERT_CMD_UPDATECRLS** - стандартное обновление всех САС, найденных в ЛСП. При выполнении данной команды перебираются все САС из ЛСП, и для каждого из них загружаются обновления из указанных в данном САС точек CDP. Для данной команды использование минимального контекста не допускается, а параметры **idat**, **odat**, **flag** не используются;

- **VCERT_CMD_UPDATECRLSCRIT** - критичное обновление всех САС, найденных в ЛСП. При выполнении данной команды перебираются все САС из ЛСП, и для каждого из них загружаются обновления из указанных в данном САС точек CDP. Если при обновлении хотя бы одного САС возникает ошибка, то ее текстовое описание возвращается в заранее выделенный блок памяти, указанный в **odat** (строка описания ошибки завершается символом '\0'). Для данной команды использование минимального контекста не допускается, а параметры **idat**, **flag** не используются. Для возможности корректного преобразования содержимого блока памяти **odat** в строковые данные необходимо удалить находящийся в конце этого блока символ '\0';

- **VCERT_CMD_SIGN_WIZARD** - запуск графической оболочки Мастера вычисления ЭП. Для данной команды использование минимального контекста и контекста проверки не допускается, а параметры **idat**, **odat**, **flag** не используются;

- **VCERT_CMD_VERIFY_WIZARD** - запуск графической оболочки Мастера проверки ЭП. Для данной команды использование минимального контекста не допускается, а параметры **idat**, **odat**, **flag** не используются;

- **VCERT_CMD_XMLREQ_WIZARD** - запуск графической оболочки Мастера формирования запроса в формате PKCS#10 или генерация запроса в формате PKCS#10 на основании существующего XML шаблона (см. описание в подразделе 1.30). Данные существующего XML шаблона берутся из блока памяти, указанного в **idat**, а сформированный запрос в формате PKCS#10 возвращается в блок памяти, указанный в **odat**. Для данной команды параметр **flag** может быть маской (побитовым ИЛИ) следующих флагов:

- **FLAG_CTRL_XMLREQ_FOR_ENCRYPTION** - выполнить генерацию закрытого ключа, предназначенного для шифрования;

- **FLAG_CTRL_XMLREQ_GOST_R_34_10_12_256** - выполнить генерацию закрытого ключа для сертификата в квалифицированном формате по ГОСТ Р 34.10-2012 (256 бит);

- **FLAG_CTRL_XMLREQ_GOST_R_34_10_12_512** - выполнить генерацию закрытого ключа для сертификата в квалифицированном формате по ГОСТ Р 34.10-2012 (512 бит);

- **FLAG_CTRL_XMLREQ_CARRIER_GENERATION** - при генерации неизвлекаемого закрытого ключа на функциональном ключевом носителе (ФКН) vdToken формировать этот ключ внутри ФКН с использованием внутреннего ДСЧ последнего;

- **VCERT_CMD_PROFILE_WIZARD** - запуск графической оболочки Мастера управления профилями пользователя. Для данной команды параметр

idat не используется. При задании параметра **odat**, в случае успешного завершения работы Мастера, в него записывается строка с именем выбранного профиля пользователя. Для данной команды параметр **flag** может быть маской (побитовым ИЛИ) следующих флагов:

- **FLAG_CTRL_PROFWIZ_PROHIBIT_MODIFIES** - разрешить только чтение существующих профилей пользователя, без возможности их создания, модификации и удаления;

- **VCERT_CMD_PRIVKEY_WIZARD** - запуск графической оболочки Мастера управления закрытыми ключами (в настоящее время реализовано только удаление закрытых ключей). Для данной команды параметры **hWnd**, **idat**, **odat**, **flag** не используются;

- **VCERT_CMD_CARRIER_SETPIN** - запуск графического интерфейса смены ПИН-кода ключевых носителей. Для данной команды параметры **hWnd**, **idat**, **odat**, **flag** не используются;

- **VCERT_CMD_CARRIER_FORMAT** - запуск графического интерфейса форматирования ключевых носителей. Для данной команды параметры **hWnd**, **idat**, **odat**, **flag** не используются;

- **idat** (in) блок памяти с входными данными (зависит от выполняемой команды);

- **odat** (out) блок памяти с выходными данными (зависит от выполняемой команды);

- **flag** (in) маска (побитовое ИЛИ) флагов выполнения (зависят от выполняемой команды).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int VCERT_UpdateCRLs ()

Функция 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**.

int VCERT_CreateXmlRequest (int flag, certificate_t reqtmpl, mem_blk_t xmlreq)

Функция создания парных ключей и XML запроса по шаблону сертификата

Аргументы:

- **flag** (in) маска (побитовое ИЛИ) флагов формирования XML-запроса (флаги **FLAG_XML_REQUEST_XXX**);

- **reqtmpl** (in) структура сертификата - шаблон, на основании которого формируется XML-запрос. Для формирования XML-запроса используются следующие поля структуры:

- X.500-имя владельца сертификата;
- альтернативное имя владельца сертификата;
- регламенты использования сертификата;

- расширенные области использования открытого ключа;
 - разрешенные области использования открытого ключа;
 - дополнения (расширения) сертификата;
- **xmlreq** (out) блок памяти с сформированным XML-запросом.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.12 Справочники и профили пользователя

1.12.1 Функции справочников и профилей пользователя

int **VCERT_CreateCertificateStore** (mem_blk_t[] certs, int ncert, mem_blk_t[] crls, int ncrl, String pstore, String localstore, int flag)

Функция формирования ПСП и ЛСП из отдельных сертификатов и САС

Аргументы:

- **certs** (in) массив 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;
- **ncert** (in) длина массива блоков памяти, содержащих сертификаты (должно выполняться условие **ncert > 0**);
- **crls** (in) массив 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;
- **ncrl** (in) длина массива блоков памяти, содержащих САС (должно выполняться условие **ncrl ≥ 0**);
- **pstore** (out) путь (URI) к создаваемому ПСП. В ПСП будут добавлены сертификаты корневых ЦС. Указанный ПСП, в случае не расположения в системном хранилище ОС Microsoft Windows, будет сформирован заново, и все находившиеся в нем ранее объекты будут удалены;
- **localstore** (out) путь (URI) к создаваемому ЛСП. В ЛСП будут добавлены остальные сертификаты и САС. Указанный ЛСП, в случае расположения в БД GDBM, будет сформирован заново, и все находившиеся в нем ранее объекты будут удалены;
- **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CreateRegistryProfile** (String profile, String storepath, int flag)

Функция создания или обновления профиля по пути к файлам ПСП и ЛСП

Аргументы:

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

– **storepath** (in) путь к файлам ПСП **local.pse** и ЛСП **local.gdbm**. Данную функцию можно использовать только в тех случаях, когда ПСП не расположен в системном хранилище ОС Microsoft Windows, а ЛСП расположен в БД GDBM;

– **flag** (in) маска (побитовое ИЛИ) флагов добавления или обновления профиля:

- **FLAG_CREATE_REGISTRY_PROFILE_OVER** - разрешить обновить (перезаписать) данные профиля в том случае, если профиль пользователя с именем **profile** уже существует.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CreateRegistryProfileEx** (String profile, String pstore, String localstore, String ldapstore, int flag)

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

Аргументы:

– **profile** (in) имя создаваемого или обновляемого профиля пользователя, например **По умолчанию**;

– **pstore** (in) путь (URI) к ПСП профиля пользователя;

– **localstore** (in) путь (URI) к ЛСП профиля пользователя;

– **ldapstore** (in) путь (URI) (опциональный) к ССС профиля пользователя;

– **flag** (in) маска (побитовое ИЛИ) флагов добавления или обновления профиля:

- **FLAG_CREATE_REGISTRY_PROFILE_OVER** - разрешить обновить (перезаписать) данные профиля в том случае, если профиль пользователя с именем **profile** уже существует.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_QueryRegistryProfile** (String[] profile, int length, int index)

Функция получения информации о профиле по индексу

Аргументы:

– **profile** (out) буфер для имени профиля - строки, заканчивающейся символом '\0' (параметр должен представлять собой массив с одним элементом);

– **length** (in) максимальная длина буфера для имени профиля;

– **index** (in) индекс профиля (должно выполняться условие **index** ≥ 0).

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_QueryRegistryProfileEx** (mem_blk_t profile, mem_blk_t pstore, mem_blk_t localstore, mem_blk_t ldapstore, int index)

Расширенная функция получения информации о профиле по индексу

Аргументы:

- **profile** (out) блок памяти для записи имени профиля - строки, заканчивающейся символом '\0';
- **psestore** (out) блок памяти для записи пути (URI) к ПСП - строки, заканчивающейся символом '\0';
- **localstore** (out) блок памяти для записи пути (URI) к ЛСП - строки, заканчивающейся символом '\0';
- **ldapstore** (out) блок памяти для записи пути (URI) к ССС - строки, заканчивающейся символом '\0';
- **index** (in) индекс профиля (должно выполняться условие **index** ≥ 0).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

Для возможности корректного преобразования содержимого блоков памяти **profile**, **psestore**, **localstore** и **ldapstore** в строковые данные необходимо удалить находящийся в конце этих блоков символ '\0'.

1.13 Визуализация объектов СУС**1.13.1 Функции визуализации объектов СУС**

int **VCERT_ShowCertificate** (mem_blk_t cert)

Функция визуализации сертификата

Аргументы:

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

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_ShowCertificateEx** (long hWnd, mem_blk_t cert)

Расширенная функция визуализации сертификата

Аргументы:

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

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_ShowCrl** (mem_blk_t crl)

Функция визуализации САС

Аргументы:

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

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_ShowCrlEx** (long hWnd, mem_blk_t crl)

Расширенная функция визуализации САС

Аргументы:

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

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_ShowRequestEx** (long hWnd, mem_blk_t req)

Расширенная функция визуализации PKCS#10 запроса

Аргументы:

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

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_ShowRevocationEx** (long hWnd, mem_blk_t rev)

Расширенная функция визуализации запроса на аннулирование сертификата

Аргументы:

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

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.14 Экспорт и импорт объектов СУС

Функции экспорта предназначены для формирования подписанного запроса в формате PKCS#10 при плановой смене рабочего сертификата пользователя, подписанного запроса на аннулирование рабочего сертификата при компрометации закрытого ключа пользователя, а также для экспорта сертификатов и САС из ПСП и ЛСП в системное хранилище ОС Microsoft Windows.

Функции импорта предназначены для добавления объектов СУС в DER-кодировке или PEM-формате в ЛСП или ССС, а также для добавления объектов СУС, содержащихся в подписанных обновлениях от ЦС и ЦР, в ПСП и ЛСП.

1.14.1 Структуры экспорта и импорта объектов СУС

Структура export_param_t

Параметры формирования запроса PKCS#10 или запроса на аннулирование:

– int **flag**

Поле с маской (побитовым ИЛИ) флагов FLAG_EXPORT_XXX формирования запроса PKCS#10 или запроса на аннулирование (должен быть установлен один

и только один из флагов **FLAG_EXPORT_PKCS10REQUEST**, **FLAG_EXPORT_REVOKEREQUEST**).

– certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

Структура import_param_t

Параметры добавления объектов СУС в ПСП, ЛСП или ССС:

– int **flag**

Поле с маской (побитовым ИЛИ) флагов **FLAG_IMPORT_XXX** добавления объектов СУС в ПСП, ЛСП или ССС (должен быть установлен один и только один из флагов **FLAG_IMPORT_CERTDERPEMFORMAT**, **FLAG_IMPORT_CRLDERPEMFORMAT**, **FLAG_IMPORT_CERTWITHPRIVATE**, **FLAG_IMPORT_SIGNEDCARAUPDATE**).

– certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

1.14.2 Функции экспорта и импорта объектов СУС

int **VCERT_ExportMem** (export_param_t pExportPara, mem_blk_t oexp)

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

Аргументы:

– **pExportPara** (in) структура параметров формирования запроса PKCS#10/на аннулирование;

– **oexp** (out) блок данных с сформированным запросом в DER-кодировке.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_ExportFile** (export_param_t pExportPara, String oexp)

Функция формирования файла с запросом PKCS#10/на аннулирование

Аргументы:

– **pExportPara** (in) структура параметров формирования запроса PKCS#10/на аннулирование;

– **oexp** (out) файл с сформированным запросом в DER-кодировке.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_ExportToSystemStore** (int flag)

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

Аргументы:

– **flag** (in) маска (побитовое ИЛИ) флагов экспорта сертификатов и САС:

• **FLAG_EXPORT_SYSTEM_STORE_MACHINE** - выполнить экспорт сертификатов и САС в системное хранилище компьютера ОС Microsoft Windows (для этого необходимы права локального администратора). Если

данный флаг не установлен, то экспорт выполняется в системное хранилище пользователя ОС Microsoft Windows;

• **FLAG_EXPORT_SYSTEM_STORE_IGNORE** - игнорировать любые ошибки, возникающие при добавлении сертификатов и САС в системное хранилище ОС Microsoft Windows.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int VCERT_ImportMem (import_param_t pImportPara, mem_blk_t iimp)

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

Аргументы:

– **pImportPara** (in) структура параметров добавления объектов СУС;

– **iimp** (in) блок данных с добавляемыми объектами в DER-кодировке или PEM-формате.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int VCERT_ImportFile (import_param_t pImportPara, String iimp)

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

Аргументы:

– **pImportPara** (in) структура параметров добавления объектов СУС;

– **iimp** (in) файл с добавляемыми объектами в DER-кодировке или PEM-формате.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.14.3 Структуры разбора и получения информации об объектах СУС

Структура othername_t

Другое имя альтернативного имени:

– String **oid**

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

– String **name**

Строка с текстовыми данными другого имени.

Структура altname_ex_t

Другие имена альтернативного имени:

– int **othername_num**

Количество элементов массива других имен альтернативного имени.

– othername_t[] **othernames**

Поле с массивом других имен альтернативного имени.

Структура `basic_constraints_ex_t`Базовые ограничения сертификата:

- `int ca`

Признак сертификата ЦС. Если значение $\neq 0$, то сертификат является сертификатом ЦС.

- `int pathlen`

Максимальная длина цепочки сертификации. Значение < 0 означает отсутствие ограничения на длину цепочки.

1.14.4 Функции разбора и получения информации об объектах СУС

`int VCERT_ParseCert (mem_blk_t icert, int info, certificate_t ocert)`

Функция разбора и получения информации о сертификате

Аргументы:

- ***icert*** (in) блок памяти с сертификатом в DER-кодировке или PEM-формате;
- ***info*** (in) требуемая возвращаемая информация о сертификате;
- ***ocert*** (out) структура разобранного сертификата.

Возвращаемые значения:

- ***VCERT_OK*** в случае успеха или ненулевой код ошибки.

`int VCERT_ParseCrl (mem_blk_t icrl, int info, crl_t ocrl)`

Функция разбора и получения информации о САС

Аргументы:

- ***icrl*** (in) блок памяти с САС в DER-кодировке или PEM-формате;
- ***info*** (in) требуемая возвращаемая информация о САС;
- ***ocrl*** (out) структура разобранного САС.

Возвращаемые значения:

- ***VCERT_OK*** в случае успеха или ненулевой код ошибки.

`int VCERT_ParseAltnameEx (mem_blk_t ialtname, altname_ex_t oaltname)`

Функция разбора и получения информации об альтернативном имени

Аргументы:

- ***ialtname*** (in) блок памяти с данными дополнения альтернативного имени в DER-кодировке;
- ***oaltname*** (out) структура разобранного альтернативного имени.

Возвращаемые значения:

- ***VCERT_OK*** в случае успеха или ненулевой код ошибки.

`int VCERT_ParseBasicConstraintsEx (mem_blk_t ibc, basic_constraints_ex_t obc)`

Функция разбора и получения информации о базовых ограничениях сертификата

Аргументы:

- **ibc** (in) блок памяти с данными дополнения базовых ограничений сертификата в DER-кодировке;
- **obc** (out) структура разобранных базовых ограничений.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_ParseKeyIdentifierEx** (mem_blk_t ikid, mem_blk_t okid)

Функция разбора и получения информации об идентификаторе ключа издателя

Аргументы:

- **ikid** (in) блок памяти с данными дополнения идентификатора ключа издателя в DER-кодировке;
- **okid** (out) блок памяти с разобранным идентификатором ключа издателя.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.15 Построение и проверка цепочек объектов СУС

1.15.1 Структуры построения и проверки цепочек объектов СУС

Структура **verify_policy_param_t**

Структура параметров построения и проверки цепочки сертификата или САС:

- int **flag**

Поле с маской (побитовым ИЛИ) флагов построения и проверки цепочки сертификата или САС:

- **FLAG_POLICY_DONOTCHECKTIMES** - не проверять сроки действия сертификата или САС, для которого строится и проверяется цепочка, а также сертификатов ЦС и САС из его цепочки;

- **FLAG_POLICY_DOCRLVERIFICATION** - строить и проверять цепочку САС. Если данный флаг не установлен, то строить и проверять цепочку сертификата;

- **FLAG_POLICY_DONOTCHECKKEYTIME** - не проверять срок действия закрытого ключ сертификата, для которого строится и проверяется цепочка.

Примечание — Указанные флаги допускается использовать только для проверки архивных сообщений.

- certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

- long **check_timeEx**

Поле с моментом времени, на который строится и проверяется цепочка сертификата или САС.

- int **keyusage**

Поле с маской (побитовым ИЛИ) разрешенных областей использования открытого ключа. Если значение не равно 0, то будет выполняться проверка

наличия заданных разрешенных областей использования открытого ключа в сертификате, для которого строится и проверяется цепочка.

– int **eku_num**

Количество элементов массива расширенных областей использования открытого ключа.

– extkeyusage_t[] **extkeyusages**

Поле с массивом расширенных областей использования открытого ключа. Если массив не пуст, то будет выполняться проверка наличия заданных расширенных областей использования открытого ключа в сертификате, для которого строится и проверяется цепочка.

– int **policy_num**

Количество элементов массива регламентов использования сертификата.

– policy_t[] **policies**

Поле с массивом регламентов использования сертификата. Если массив не пуст, то будет выполняться проверка наличия заданных регламентов использования сертификата в сертификате, для которого строится и проверяется цепочка.

1.15.2 Функции построения и проверки цепочек объектов СУС

int **VCERT_VerifyCert** (verify_param_t pVerifyPara, mem_blk_t icert, certificate_t ocert) Функция построения и проверки цепочки сертификата

Аргументы:

- **pVerifyPara** (in) структура параметров проверки сертификата. Проверка сертификата выполняется на текущий момент времени;
- **icert** (in) блок памяти с проверяемым сертификатом в DER-кодировке или PEM-формате;
- **ocert** (out) структура разобранного сертификата.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_VerifyCertificatePolicy** (verify_policy_param_t pPolicyPara, mem_blk_t icert) Расширенная функция построения и проверки цепочки сертификата

Аргументы:

- **pPolicyPara** (in) структура параметров построения и проверки цепочки сертификата;
- **icert** (in) блок памяти с проверяемым сертификатом в DER-кодировке или PEM-формате.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_VerifyCrlPolicy** (verify_policy_param_t pPolicyPara, mem_blk_t icrl) Расширенная функция построения и проверки цепочки САС

Аргументы:

- **pPolicyPara** (in) структура параметров построения и проверки цепочки САС (необходимо установить флаг **FLAG_POLICY_DOCRLVERIFICATION**);
- **icrl** (in) блок памяти с проверяемым САС в DER-кодировке или PEM-формате.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.16 Вычисление хэш-значений

1.16.1 Функции вычисления хэш-значений

int **VCERT_BlkJHashMem** (String algorithm, mem_blk_t data, mem_blk_t hash)

Функция блочного вычисления хэш-значения блока памяти

Аргументы:

- **algorithm** (in) 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** (in) блок памяти с исходными данными (блок памяти может быть нулевой длины);

- **hash** (out) блок памяти для вычисленного хэш-значения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_BlkJHashFile** (String algorithm, String data, mem_blk_t hash)

Функция блочного вычисления хэш-значения файла

Аргументы:

- **algorithm** (in) 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** (in) имя файла с исходными данными (файл не может быть нулевой длины);

- **hash** (out) блок памяти для вычисленного хэш-значения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_StrHashInitMem** (String algorithm, strhash_handle_t hStr)

Функция инициализации потокового вычисления хэш-значения блоков памяти

Аргументы:

– **algorithm** (in) 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);

– **hStr** (out) контекст выполнения потоковой операции.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_StrHashUpdateMem** (strhash_handle_t hStr, mem_blk_t data)

Функция продолжения потокового вычисления хэш-значения блоков памяти

Аргументы:

– **hStr** (in) контекст выполнения потоковой операции;

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

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_StrHashFinalMem** (strhash_handle_t hStr, mem_blk_t hash)

Функция финализации потокового вычисления хэш-значения блоков памяти

Аргументы:

– **hStr** (in) контекст выполнения потоковой операции;

– **hash** (out) блок памяти для вычисленного хэш-значения.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_StrHashFile** (String algorithm, String data, mem_blk_t hash)

Функция потокового вычисления хэш-значения файла

Аргументы:

– **algorithm** (in) 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** (in) имя файла с исходными данными (файл не может быть нулевой длины);

– **hash** (out) блок памяти для вычисленного хэш-значения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.17 Вычисление и проверка ЭП хэш-значений**1.17.1 Функции вычисления и проверки ЭП хэш-значений**

int **VCERT_SignHashMem** (certid_t mycert, mem_blk_t hash, mem_blk_t sign)

Функция вычисления ЭП хэш-значения

Аргументы:

- **mycert** (in) структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ). Вычисление ЭП выполняется на ключе ЭП, соответствующем указанному сертификату;
- **hash** (in) блок памяти с хэш-значением;
- **sign** (out) блок памяти для вычисленной ЭП хэш-значения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_VerifyHashMem** (certid_t mycert, mem_blk_t sender, mem_blk_t hash, mem_blk_t sign)

Функция проверки ЭП хэш-значения

Аргументы:

- **mycert** (in) структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ);
- **sender** (in) блок памяти с сертификатом, на котором необходимо проверять ЭП (в DER-кодировке или PEM-формате);
- **hash** (in) блок памяти с хэш-значением;
- **sign** (in) блок памяти с проверяемой ЭП хэш-значения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.18 Поиск и перечисление объектов СУС**1.18.1 Структуры поиска сертификатов****Структура find_param_t**

Структура параметров поиска сертификатов по заданному шаблону:

- int **flag**

Поле с маской (побитовым ИЛИ) флагов поиска сертификатов (FLAG_FIND_XXX).

- certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

- `certificate_t` **certTemplate**

Поле со структурой шаблона сертификата для выполнения поиска.

- `int` **info**

Поле с требуемой возвращаемой информацией о найденных сертификатах.

*Примечание - Если не задан ни один из флагов **FLAG_FIND_DONOTCHECKTIMES**, **FLAG_FIND_DONOTVERIFYCHAIN**, **FLAG_FIND_DONOTCHECKKEYTIME**, то функция поиска сертификатов возвращает все найденные и действительные - по построению и проверке цепочки, по сроку действия сертификата, по сроку действия закрытого ключа - сертификаты.*

Структура `find_result_t`

Структура результатов поиска сертификатов по заданному шаблону:

- `int` **num**

Количество элементов массива найденных сертификатов.

- `certificate_t[]` **certs**

Поле с массивом найденных сертификатов.

1.18.2 Функции поиска сертификатов

`int VCERT_FindCert (find_param_t pFindPara, find_result_t pFindResult)`

Функция поиска сертификатов по заданному шаблону

Аргументы:

- **pFindPara** (in) структура параметров поиска сертификатов по заданному шаблону;
- **pFindResult** (out) структура результата поиска сертификатов по заданному шаблону.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки. При поиске по уникальному критерию данная функция может вернуть код ошибки, возникшей при построении и проверке цепочки найденного сертификата.

1.18.3 Функции перечисления объектов СУС

`int VCERT_EnumStoreObjects (certid_t mycert, enuobj_handle_t hEnu, mem_blk_t object, int flag)`

Функция перечисления объектов справочников

Аргументы:

- **mycert** (in) структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ);
- **hEnu** (out) контекст перечисления объектов справочников (при первом вызове значение контекста должно быть установлено в NULL);
- **object** (out) блок памяти с очередным объектом в DER-кодировке;
- **flag** (in) маска (побитовое ИЛИ) флагов перечисления (разрешено задавать только один из флагов **FLAG_ENUM_STORE_OBJECTS_STORE_XXX** и только один из флагов **FLAG_ENUM_STORE_OBJECTS_OBJECT_XXX**).

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки. Для полного перечисления объектов справочника функцию следует вызывать до тех пор, пока возвращаемое ею значение равно **VCERT_OK**.

int **VCERT_EnumStoreObjectsEx** (certid_t mycert, String suri, String lqry, enuobj_handle_t hEnu, mem_blk_t object, int flag)

Расширенная функция перечисления объектов справочников

Аргументы:

– **mycert** (in) структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ);

– **suri** (in) URI справочника для перечисления объектов (значение параметра используется при заданном флаге **FLAG_ENUM_STORE_OBJECTS_STORE_SURI**);

– **lqry** (in) логическое выражение вида **(&(vdSubjName=*Иван*)(vdKeyId=1234*))**, позволяющее ограничить просматриваемые в CCC контейнеры. Данный аргумент разрешено задавать только при переборе объектов в CCC с указанием одного из флагов **FLAG_ENUM_STORE_OBJECTS_STORE_LDAP** или **FLAG_ENUM_STORE_OBJECTS_STORE_SURI**;

– **hEnu** (out) контекст перечисления объектов справочников (при первом вызове значение контекста должно быть установлено в NULL);

– **object** (out) блок памяти с очередным объектом в DER-кодировке;

– **flag** (in) маска (побитовое ИЛИ) флагов перечисления (разрешено задавать только один из флагов **FLAG_ENUM_STORE_OBJECTS_STORE_XXX** и только один из флагов **FLAG_ENUM_STORE_OBJECTS_OBJECT_XXX**).

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки. Для полного перечисления объектов справочника функцию следует вызывать до тех пор, пока возвращаемое ею значение равно **VCERT_OK**.

1.19 Вычисление ЭП CMS-сообщений

Подробное описание типов подписанных CMS-сообщений приводится в документе ВАНБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

Следует учитывать тот факт, что при добавлении ЭП к подписанному CMS-сообщению необходимо использовать те функции вычисления ЭП, которые не изменяют ни тип подписанного сообщения - с совмещенной(ыми) или с отдельной(ыми) ЭП, ни тип ASN.1 кодировки - определенной или неопределенной длины.

1.19.1 Структуры вычисления ЭП CMS-сообщений**Структура sign_param_t**

Структура параметров вычисления ЭП CMS-сообщений:

– int **flag**

Поле с маской (побитовым ИЛИ) флагов вычисления ЭП CMS-сообщений (FLAG_CMS_XXX).

– certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

1.19.2 Функции блочного вычисления совмещенной ЭП CMS-сообщений

int **VCERT_CmsBlkAttSignMem** (sign_param_t pSignPara, mem_blk_t data, mem_blk_t ocms)

Функция блочного вычисления совмещенной ЭП блока памяти

Аргументы:

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений;
- **data** (in) блок памяти с данными для вычисления ЭП (должно выполняться условие **data->len > 0**);
- **ocms** (out) блок памяти с подписанным CMS-сообщением.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsBlkAttSignFile** (sign_param_t pSignPara, String data, String ocms)

Функция блочного вычисления совмещенной ЭП файла

Аргументы:

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений;
- **data** (in) файл (не нулевой длины) с данными для вычисления ЭП;
- **ocms** (out) файл с подписанным CMS-сообщением.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.19.3 Функции потокового вычисления совмещенной ЭП CMS-сообщений

int **VCERT_CmsStrAttSignInitMem** (sign_param_t pSignPara, mem_blk_t data, strcms_handle_t hStr)

Функция инициализации потокового вычисления совмещенной ЭП блока памяти

Аргументы:

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **data** (in) блок памяти с началом данных для вычисления ЭП (должно выполняться условие **data->len > 0**). При добавлении ЭП к подписанному CMS-сообщению должно выполняться условие **data->len ≥ 128**;
- **hStr** (out) контекст потоковой операции.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrAttSignUpdateMem** (sign_param_t pSignPara, strcms_handle_t hStr, mem_blk_t data, mem_blk_t ocms)

Функция продолжения потокового вычисления совмещенной ЭП блока памяти
Аргументы:

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **data** (in) блок памяти с продолжением данных для вычисления ЭП;
- **ocms** (out) блок памяти с продолжением подписанного CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrAttSignFinalMem** (sign_param_t pSignPara, strcms_handle_t hStr, mem_blk_t ocms)

Функция финализации потокового вычисления совмещенной ЭП блока памяти
Аргументы:

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **ocms** (out) блок памяти с окончанием подписанного CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrAttSignFile** (sign_param_t pSignPara, String data, String ocms)

Функция потокового вычисления совмещенной ЭП файла
Аргументы:

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений;
- **data** (in) файл (не нулевой длины) с данными для вычисления ЭП;
- **ocms** (out) файл с подписанным CMS-сообщением.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.19.4 Функции блочного вычисления отдельной ЭП CMS-сообщений

int **VCERT_CmsBlkDetSignMem** (sign_param_t pSignPara, mem_blk_t data, mem_blk_t icms, mem_blk_t ocms)

Функция блочного вычисления отделенной ЭП блока памяти**Аргументы:**

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений;
- **data** (in) блок памяти с данными для вычисления ЭП (должно выполняться условие **data->len > 0**);
- **icms** (in) блок памяти (опциональный) с подписанным CMS-сообщением для добавления ЭП;
- **ocms** (out) блок памяти с подписанным CMS-сообщением.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsBlkDetSignFile** (sign_param_t pSignPara, String data, String icms, String ocms)

Функция блочного вычисления отделенной ЭП файла**Аргументы:**

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений;
- **data** (in) файл (не нулевой длины) с данными для вычисления ЭП;
- **icms** (in) файл (опциональный) с подписанным CMS-сообщением для добавления ЭП;
- **ocms** (out) файл с подписанным CMS-сообщением.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.19.5 Функции потокового вычисления отделенной ЭП CMS-сообщений

int **VCERT_CmsStrDetSignInitMem** (sign_param_t pSignPara, mem_blk_t icms, strcms_handle_t hStr)

Функция инициализации потокового вычисления отделенной ЭП блока памяти**Аргументы:**

- **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **icms** (in) блок памяти (опциональный) с подписанным CMS-сообщением для добавления ЭП;
- **hStr** (out) контекст потоковой операции.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrDetSignUpdateMem** (sign_param_t pSignPara, strcms_handle_t hStr, mem_blk_t data)

Функция продолжения потокового вычисления отделенной ЭП блока памяти**Аргументы:**

– **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);

– **hStr** (in) контекст потоковой операции;

– **data** (in) блок памяти с продолжением данных для вычисления ЭП.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrDetSignFinalMem** (sign_param_t pSignPara, strcms_handle_t hStr, mem_blk_t ocms)

Функция финализации потокового вычисления отделенной ЭП блока памяти

Аргументы:

– **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);

– **hStr** (in) контекст потоковой операции;

– **ocms** (out) блок памяти с подписанным CMS-сообщением.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrDetSignFile** (sign_param_t pSignPara, String data, String icms, String ocms)

Функция потокового вычисления отделенной ЭП файла

Аргументы:

– **pSignPara** (in) структура параметров вычисления ЭП CMS-сообщений;

– **data** (in) файл (не нулевой длины) с данными для вычисления ЭП;

– **icms** (in) файл (опциональный) с подписанным CMS-сообщением для добавления ЭП;

– **ocms** (out) файл с подписанным CMS-сообщением.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.20 Проверка ЭП CMS-сообщений

Подробное описание процедуры проверки ЭП CMS-сообщений приводится в документе ВАМБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

1.20.1 Структуры проверки ЭП CMS-сообщений

Структура verify_param_t

Структура параметров проверки ЭП CMS-сообщений:

– int **flag**

Поле с маской (побитовым ИЛИ) флагов проверки ЭП CMS-сообщений (**FLAG_CMS_VERIFY_XXX**).

– **certid_t mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть **null** при использовании локального Средства КЗИ).

– **keyusage_t keyUsage**

Поле с маской (побитовым ИЛИ) разрешенных областей использования открытого ключа.

– **int policy_num**

Количество элементов массива регламентов использования сертификата.

– **policy_t[] policies**

Поле с массивом регламентов использования сертификата.

– **int extkeyusage_num**

Количество элементов массива расширенных областей использования открытого ключа.

– **extkeyusage_t[] extKeyUsage**

Поле с массивом расширенных областей использования открытого ключа.

– **int nSignToDelete**

Поле с количеством ЭП, которые необходимо удалить, начиная с конца CMS-сообщения. Для удаления всех ЭП следует подать значение, равное **DELETE_ALL_SIGNS**. При проверке совмещенных потоковых ЭП разрешено подавать только значения **0** и **DELETE_ALL_SIGNS**.

– **int minsigns**

Поле с минимально допустимым количеством ЭП в CMS-сообщении.

– **int info**

Поле с требуемой возвращаемой информацией о сертификатах подписантов.

Структура **sign_status_t**

Структура результата проверки конкретной ЭП CMS-сообщения:

– **error_status_t status**

Поле с результатом проверки конкретной ЭП:

- **== VCERT_OK** - ЭП проверена успешно;
- **!= VCERT_OK** - код ошибки проверки ЭП.

– **long timeEx**

Поле со временем вычисления ЭП в часовом поясе UTC, взятое из аутентифицируемого атрибута **rsaSigningTime** данной ЭП (равно **0** в случае отсутствия указанного атрибута в ЭП).

– **certificate_t cert**

Поле со структурой сертификата, на котором выполнялась проверка ЭП, т.е. сертификата отправителя данной ЭП (равно **NULL** в случае ненахождения сертификата отправителя).

Структура **verify_result_t**

Структура результатов проверки ЭП CMS-сообщений:

– int **sign_num**

Количество элементов массива результатов проверки ЭП подписанного CMS-сообщения.

– sign_status_t[] **signs**

Поле с массивом результатов проверки ЭП подписанного CMS-сообщения.

1.20.2 Функции блочной проверки совмещенных ЭП CMS-сообщений

int **VCERT_CmsBlkAttVerifyMem** (verify_param_t pVerifyPara, mem_blk_t icms, mem_blk_t data, verify_result_t pVerifyResult)

Функция блочной проверки совмещенных ЭП блока памяти

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений;
- **icms** (in) блок памяти с подписанным CMS-сообщением (должно выполняться условие **icms->len > 0**);
- **data** (out) блок памяти с выходными данными. Заполняется при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **NULL**;
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успешной проверки всех ЭП (необходимо анализировать структуру **pVerifyResult**);
- **VCERT_E_VERIFY** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать структуру **pVerifyResult**);
- **!= VCERT_OK && != VCERT_E_VERIFY** в случае другой ошибки (запрещено анализировать структуру **pVerifyResult**).

int **VCERT_CmsBlkAttVerifyFile** (verify_param_t pVerifyPara, String icms, String data, verify_result_t pVerifyResult)

Функция блочной проверки совмещенных ЭП файла

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением;
- **data** (out) файл с выходными данными. Создается при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **NULL**;
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успешной проверки всех ЭП (необходимо анализировать структуру **pVerifyResult**);
- **VCERT_E_VERIFY** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать структуру **pVerifyResult**);

– ***!= VCERT_OK && != VCERT_E_VERIFY*** в случае другой ошибки (запрещено анализировать структуру **pVerifyResult**).

1.20.3 Функции потоковой проверки совмещенных ЭП CMS-сообщений

int **VCERT_CmsStrAttVerifyInitMem** (verify_param_t pVerifyPara, mem_blk_t icms, strcms_handle_t hStr)

Функция инициализации потоковой проверки совмещенных ЭП блока памяти

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **icms** (in) блок памяти с началом подписанного CMS-сообщения (должно выполняться условие **icms->len ≥ 128**);
- **hStr** (out) контекст потоковой операции.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrAttVerifyUpdateMem** (verify_param_t pVerifyPara, strcms_handle_t hStr, mem_blk_t icms, mem_blk_t data)

Функция продолжения потоковой проверки совмещенных ЭП блока памяти

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **icms** (in) блок памяти с продолжением подписанного CMS-сообщения;
- **data** (out) блок памяти с продолжением выходных данных. Заполняется при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES**, иначе может быть равен **NULL**.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrAttVerifyFinalMem** (verify_param_t pVerifyPara, strcms_handle_t hStr, mem_blk_t data, verify_result_t pVerifyResult)

Функция финализации потоковой проверки совмещенных ЭП блока памяти

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **data** (out) блок памяти с окончанием выходных данных. Заполняется при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES**, иначе

может быть равен **NULL**;

- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успешной проверки всех ЭП (необходимо анализировать структуру **pVerifyResult**);
- **VCERT_E_VERIFY** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать структуру **pVerifyResult**);
- **!= VCERT_OK && != VCERT_E_VERIFY** в случае другой ошибки (запрещено анализировать структуру **pVerifyResult**).

int VCERT_CmsStrAttVerifyFile (verify_param_t pVerifyPara, String icms, String data, verify_result_t pVerifyResult)

Функция потоковой проверки совмещенных ЭП файла

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением;
- **data** (out) файл с выходными данными. Создается при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **NULL**;
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успешной проверки всех ЭП (необходимо анализировать структуру **pVerifyResult**);
- **VCERT_E_VERIFY** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать структуру **pVerifyResult**);
- **!= VCERT_OK && != VCERT_E_VERIFY** в случае другой ошибки (запрещено анализировать структуру **pVerifyResult**).

1.20.4 Функции блочной проверки отделенных ЭП CMS-сообщений

int VCERT_CmsBlkDetVerifyMem (verify_param_t pVerifyPara, mem_blk_t data, mem_blk_t icms, mem_blk_t ocms, verify_result_t pVerifyResult)

Функция блочной проверки отделенных ЭП блока памяти

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений;
- **data** (in) блок памяти с данными для проверки ЭП (должно выполняться условие **data->len > 0**);
- **icms** (in) блок памяти с подписанным CMS-сообщением (должно выполняться условие **icms->len > 0**);
- **ocms** (out) блок памяти с выходным CMS-сообщением. Заполняется при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **NULL**;

- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успешной проверки всех ЭП (необходимо анализировать структуру **pVerifyResult**);
- **VCERT_E_VERIFY** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать структуру **pVerifyResult**);
- **!= VCERT_OK && != VCERT_E_VERIFY** в случае другой ошибки (запрещено анализировать структуру **pVerifyResult**).

int **VCERT_CmsBlkDetVerifyFile** (verify_param_t pVerifyPara, String data, String icms, String ocms, verify_result_t pVerifyResult)

Функция блочной проверки отделенных ЭП файла

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений;
- **data** (in) файл (не нулевой длины) с данными для проверки ЭП;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением;
- **ocms** (out) файл с выходным CMS-сообщением. Создается при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **NULL**;
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успешной проверки всех ЭП (необходимо анализировать структуру **pVerifyResult**);
- **VCERT_E_VERIFY** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать структуру **pVerifyResult**);
- **!= VCERT_OK && != VCERT_E_VERIFY** в случае другой ошибки (запрещено анализировать структуру **pVerifyResult**).

1.20.5 Функции потоковой проверки отделенных ЭП CMS-сообщений

int **VCERT_CmsStrDetVerifyInitMem** (verify_param_t pVerifyPara, mem_blk_t icms, strcms_handle_t hStr)

Функция инициализации потоковой проверки отделенных ЭП блока памяти

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **icms** (in) блок памяти с подписанным CMS-сообщением (должно выполняться условие **icms->len > 0**);
- **hStr** (out) контекст потоковой операции.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrDetVerifyUpdateMem** (verify_param_t pVerifyPara, strcms_handle_t hStr, mem_blk_t data)

Функция продолжения потоковой проверки отделенных ЭП блока памяти

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **data** (in) блок памяти с продолжением данных для проверки ЭП.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrDetVerifyFinalMem** (verify_param_t pVerifyPara, strcms_handle_t hStr, mem_blk_t ocms, verify_result_t pVerifyResult)

Функция финализации потоковой проверки отделенных ЭП блока памяти

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **ocms** (out) блок памяти с выходным CMS-сообщением. Заполняется при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES** (в случае успешной проверки всех ЭП CMS-сообщения), иначе может быть равен **NULL**;
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успешной проверки всех ЭП (необходимо анализировать структуру **pVerifyResult**);
- **VCERT_E_VERIFY** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать структуру **pVerifyResult**);
- **!= VCERT_OK && != VCERT_E_VERIFY** в случае другой ошибки (запрещено анализировать структуру **pVerifyResult**).

int **VCERT_CmsStrDetVerifyFile** (verify_param_t pVerifyPara, String data, String icms, String ocms, verify_result_t pVerifyResult)

Функция потоковой проверки отделенных ЭП файла

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений;
- **data** (in) файл (не нулевой длины) с данными для проверки ЭП;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением;
- **ocms** (out) файл с выходным CMS-сообщением. Создается при установке флага проверки **FLAG_CMS_VERIFY_DELETESIGNATURES** (в случае успеш-

ной проверки всех ЭП CMS-сообщения), иначе может быть равен **NULL**;

- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успешной проверки всех ЭП (необходимо анализировать структуру **pVerifyResult**);
- **VCERT_E_VERIFY** в случае ошибки проверки хотя бы одной ЭП (необходимо анализировать структуру **pVerifyResult**);
- **!= VCERT_OK && != VCERT_E_VERIFY** в случае другой ошибки (запрещено анализировать структуру **pVerifyResult**).

1.21 Зашифрование CMS-сообщений

Подробное описание формата зашифрованных CMS-сообщений приводится в документе ВАМБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

Следует обратить внимание на то, что при потоковом зашифровании CMS-сообщения в виде блока(ов) памяти функция продолжения потокового зашифрования должна быть вызвана хотя-бы один раз, пусть и с входным блоком памяти нулевой длины.

1.21.1 Структуры зашифрования CMS-сообщений

Структура **encrypt_param_t**

Структура параметров зашифрования CMS-сообщений:

- int **flag**

Поле с маской (побитовым ИЛИ) флагов зашифрования CMS-сообщений (FLAG_CMS_ENCRYPT_XXX).

- certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

- int **receiver_num**

*Количество элементов массива шаблонов сертификатов получателей (должно выполняться условие **receiver_num > 0**).*

- certificate_t **receivers**

Поле с массивом шаблонов сертификатов получателей.

1.21.2 Функции блочного зашифрования CMS-сообщений

int **VCERT_CmsBlkEncryptMem** (encrypt_param_t pEncryptPara, mem_blk_t data, mem_blk_t ocms)

Функция блочного зашифрования блока памяти

Аргументы:

- **pEncryptPara** (in) структура параметров зашифрования CMS-сообщений;
- **data** (in) блок памяти с данными для зашифрования (должно выполняться условие **data->len > 0**);
- **ocms** (out) блок памяти с зашифрованным CMS-сообщением.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsBlkEncryptFile** (encrypt_param_t pEncryptPara, String data, String ocms)

Функция блочного зашифрования файла

Аргументы:

- **pEncryptPara** (in) структура параметров зашифрования CMS-сообщений;
- **data** (in) файл (не нулевой длины) с данными для зашифрования;
- **ocms** (out) файл с зашифрованным CMS-сообщением.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.21.3 Функции потокового зашифрования CMS-сообщений

int **VCERT_CmsStrEncryptInitMem** (encrypt_param_t pEncryptPara, mem_blk_t data, strcms_handle_t hStr)

Функция инициализации потокового зашифрования блока памяти

Аргументы:

- **pEncryptPara** (in) структура параметров зашифрования CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **data** (in) блок памяти с началом данных для зашифрования;
- **hStr** (out) контекст потоковой операции.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrEncryptUpdateMem** (encrypt_param_t pEncryptPara, strcms_handle_t hStr, mem_blk_t data, mem_blk_t ocms)

Функция продолжения потокового зашифрования блока памяти

Аргументы:

- **pEncryptPara** (in) структура параметров зашифрования CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **data** (in) блок памяти с продолжением данных для зашифрования;
- **ocms** (out) блок памяти с продолжением зашифрованного CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrEncryptFinalMem** (encrypt_param_t pEncryptPara, strcms_handle_t hStr, mem_blk_t ocms)

Функция финализации потокового зашифрования блока памяти

Аргументы:

- **pEncryptPara** (in) структура параметров зашифрования CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **ocms** (out) блок памяти с окончанием зашифрованного CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrEncryptFile** (encrypt_param_t pEncryptPara, String data, String ocms)

Функция потокового зашифрования файла

Аргументы:

- **pEncryptPara** (in) структура параметров зашифрования CMS-сообщений;
- **data** (in) файл (не нулевой длины) с данными для зашифрования;
- **ocms** (out) файл с зашифрованным CMS-сообщением.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.22 Расшифрование CMS-сообщений

Подробное описание процесса расшифрования зашифрованных CMS-сообщений приводится в документе ВАНБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

Следует обратить внимание на то, что при потоковом расшифровании CMS-сообщения в виде блока(ов) памяти функция продолжения потокового расшифрования должна быть вызвана хотя-бы один раз, пусть и с входным блоком памяти нулевой длины.

1.22.1 Структуры расшифрования CMS-сообщений

Структура decrypt_param_t

Структура параметров расшифрования CMS-сообщений:

- int **flag**

Поле с маской флагов (зарезервировано, должно быть равно 0).

- certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

- int **info**

Поле с требуемой возвращаемой информацией о сертификате отправителя.

Структура decrypt_result_t

Структура результатов расшифрования CMS-сообщений:

- certificate_t **sender**

Поле со структурой сертификата отправителя зашифрованного сообщения.

Поскольку шифрование выполняется анонимным способом, данное поле не заполняется.

1.22.2 Функции блочного расшифрования CMS-сообщений

int **VCERT_CmsBlkDecryptMem** (decrypt_param_t pDecryptPara, mem_blk_t icms, mem_blk_t data, decrypt_result_t pDecryptResult)

Функция блочного расшифрования блока памяти

Аргументы:

- **pDecryptPara** (in) структура параметров расшифрования CMS-сообщений;
- **icms** (in) блок памяти с зашифрованным CMS-сообщением (должно выполняться условие **icms->len > 0**);
- **data** (out) блок памяти с расшифрованными данными;
- **pDecryptResult** (out) структура результата расшифрования CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsBlkDecryptFile** (decrypt_param_t pDecryptPara, String icms, String data, decrypt_result_t pDecryptResult)

Функция блочного расшифрования файла

Аргументы:

- **pDecryptPara** (in) структура параметров расшифрования CMS-сообщений;
- **icms** (in) файл (не нулевой длины) с зашифрованным CMS-сообщением;
- **data** (out) файл с расшифрованными данными;
- **pDecryptResult** (out) структура результата расшифрования CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.22.3 Функции потокового расшифрования CMS-сообщений

int **VCERT_CmsStrDecryptInitMem** (decrypt_param_t pDecryptPara, mem_blk_t icms, strm_handle_t hStr)

Функция инициализации потокового расшифрования блока памяти

Аргументы:

- **pDecryptPara** (in) структура параметров расшифрования CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **icms** (in) блок памяти с началом зашифрованного CMS-сообщения (должно выполняться условие **icms->len ≥ 128**);
- **hStr** (out) контекст потоковой операции.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrDecryptUpdateMem** (decrypt_param_t pDecryptPara, strcms_handle_t hStr, mem_blk_t icms, mem_blk_t data)

Функция продолжения потокового расшифрования блока памяти

Аргументы:

– **pDecryptPara** (in) структура параметров расшифрования CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);

– **hStr** (in) контекст потоковой операции;

– **icms** (in) блок памяти с продолжением зашифрованного CMS-сообщения;

– **data** (out) блок памяти с продолжением расшифрованных данных.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrDecryptFinalMem** (decrypt_param_t pDecryptPara, strcms_handle_t hStr, mem_blk_t data, decrypt_result_t pDecryptResult)

Функция финализации потокового расшифрования блока памяти

Аргументы:

– **pDecryptPara** (in) структура параметров расшифрования CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);

– **hStr** (in) контекст потоковой операции;

– **data** (out) блок памяти с окончанием расшифрованных данных;

– **pDecryptResult** (out) структура результата расшифрования CMS-сообщения.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrDecryptFile** (decrypt_param_t pDecryptPara, String icms, String data, decrypt_result_t pDecryptResult)

Функция потокового расшифрования файла

Аргументы:

– **pDecryptPara** (in) структура параметров расшифрования CMS-сообщений;

– **icms** (in) файл (не нулевой длины) с зашифрованным CMS-сообщением;

– **data** (out) файл с расшифрованными данными;

– **pDecryptResult** (out) структура результата расшифрования CMS-сообщения.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.23 Преобразование совмещенных и отделенных ЭП

1.23.1 Функции блочного преобразования отделенных ЭП в совмещенные

int **VCERT_CmsBlkSignAttachMem** (mem_blk_t data, mem_blk_t icms, mem_blk_t ocms, int flag)

Функция блочного преобразования отделенных ЭП блока памяти в совмещенные

Аргументы:

- **data** (in) блок памяти с подписанными данными (должно выполняться условие **data->len > 0**);
- **icms** (in) блок памяти с подписанным CMS-сообщением с отделенными ЭП (должно выполняться условие **icms->len > 0**);
- **ocms** (out) блок памяти с подписанным CMS-сообщением с совмещенными ЭП;
- **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsBlkSignAttachFile** (String data, String icms, String ocms, int flag)

Функция блочного преобразования отделенных ЭП файла в совмещенные

Аргументы:

- **data** (in) файл (не нулевой длины) подписанными с данными;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением с отделенными ЭП;
- **ocms** (out) файл с подписанным CMS-сообщением с совмещенными ЭП;
- **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.23.2 Функции блочного преобразования совмещенных ЭП в отделенные

int **VCERT_CmsBlkSignDetachMem** (mem_blk_t icms, mem_blk_t data, mem_blk_t ocms, int flag)

Функция блочного преобразования совмещенных ЭП блока памяти в отделенные

Аргументы:

- **icms** (in) блок памяти с подписанным CMS-сообщением с совмещенными ЭП (должно выполняться условие **icms->len > 0**);
- **data** (out) блок памяти с подписанными данными;
- **ocms** (out) блок памяти с подписанным CMS-сообщением с отделенными ЭП;
- **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsBlkSignDetachFile** (String icms, String data, String ocms, int flag)

Функция блочного преобразования совмещенных ЭП файла в отделенные

Аргументы:

- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением с совмещенными ЭП;
- **data** (out) файл с подписанными данными;
- **ocms** (out) файл с подписанным CMS-сообщением с отделенными ЭП;
- **flag** (in) маска флагов (зарезервировано, должно быть равно 0).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.23.3 Функции потокового преобразования совмещенных ЭП в отделенные

int **VCERT_CmsStrSignDetachInitMem** (mem_blk_t icms, strcms_handle_t hStr, int flag)

Функция инициализации потокового преобраз. совмещенных ЭП блока памяти

Аргументы:

- **icms** (in) блок памяти с началом подписанного CMS-сообщения с совмещенными ЭП (должно выполняться условие **icms->len** ≥ 128);
- **hStr** (out) контекст потоковой операции;
- **flag** (in) маска флагов (зарезервировано, должно быть равно 0).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrSignDetachUpdateMem** (strcms_handle_t hStr, mem_blk_t icms, mem_blk_t data)

Функция продолжения потокового преобраз. совмещенных ЭП блока памяти

Аргументы:

- **hStr** (in) контекст потоковой операции;
- **icms** (in) блок памяти с продолжением подписанного CMS-сообщения с совмещенными ЭП;
- **data** (out) блок памяти с продолжением подписанных данных.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrSignDetachFinalMem** (strcms_handle_t hStr, mem_blk_t data, mem_blk_t ocms)

Функция финализации потокового преобраз. совмещенных ЭП блока памяти

Аргументы:

- **hStr** (in) контекст потоковой операции;
- **data** (out) блок памяти с окончанием подписанных данных;
- **ocms** (out) блок памяти с подписанным CMS-сообщением с отделенными ЭП.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrSignDetachFile** (String icms, String data, String ocms, int flag)

Функция потокового преобразования совмещенных ЭП файла в отделенные

Аргументы:

- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением с совмещенными ЭП;
- **data** (out) файл с подписанными данными;
- **ocms** (out) файл с подписанным CMS-сообщением с отделенными ЭП;
- **flag** (in) маска флагов (зарезервировано, должно быть равно 0).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.24 Получение информации о CMS-сообщениях

1.24.1 Структуры получения информации о CMS-сообщениях

Структура cms_certid_t

Структура идентификатора сертификата CMS-сообщения:

- int **type**

Поле с типом структуры идентификатора сертификатов CMS-сообщения:

- **FLAG_CMS_CERTID_TYPE_ISSN** - структура идентификатора сертификата содержит пару Имя издателя/Серийный номер сертификата;
- **FLAG_CMS_CERTID_TYPE_SKID** - структура идентификатора сертификата содержит данные дополнения "Идентификатор ключа владельца".

- String **issuer**

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

– String **subjKeyId**

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 **FLAG_CMS_CERTID_TYPE_SKID**).

Структура **cms_signinf_t**

Структура информации о подписанте CMS-сообщения:

– int **version**

Поле с номером версии структуры информации о подписанте CMS-сообщения.

– String **digestAlg**

Поле с строкой 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).

– String **signatureAlg**

Поле с строкой объектного идентификатора (OID) алгоритма ЭП:

- "1.2.643.2.2.19" - ЭП по ГОСТ Р 34.10-2001;
- "1.2.643.7.1.1.1.1" - ЭП по ГОСТ Р 34.10-2012 (256 бит);
- "1.2.643.7.1.1.1.2" - ЭП по ГОСТ Р 34.10-2012 (512 бит).

– cms_certid_t **signerId**

Поле со структурой идентификатора сертификата подписанта CMS-сообщения.

– long **signingTimeEx**

Поле со временем вычисления ЭП, взятым из аутентифицированного атрибута **PKCS#9 signingTime** (при отсутствии данного атрибута значение равно 0).

– int **authAttr_num**

Количество элементов массива строк объектных идентификаторов аутентифицированных атрибутов, находящихся в ЭП.

– policy_t[] **authAttrs**

Поле с массивом строк объектных идентификаторов аутентифицированных атрибутов, находящихся в ЭП (например, объектный идентификатор **1.2.840.113549.1.9.5** соответствует атрибуту **PKCS#9 signingTime**).

– int **unauthAttr_num**

Количество элементов массива строк объектных идентификаторов неаутентифицированных атрибутов, находящихся в ЭП.

– policy_t[] **unauthAttrs**

Поле с массивом строк объектных идентификаторов неаутентифицированных атрибутов, находящихся в ЭП (например, объектный идентификатор **1.2.840.113549.1.9.16.1.4** соответствует атрибуту **RFC 3161**).

timeStampToken).

Структура cms_recinf_t

Структура информации о получателе CMS-сообщения:

– int **type**

Поле с типом структуры информации о получателе CMS-сообщения:

- **FLAG_CMS_RECINF_TYPE_KTRI** - структура информации о получателе типа *KeyTransport*;
- **FLAG_CMS_RECINF_TYPE_KARI** - структура информации о получателе типа *KeyAgreement*.

– int **index**

*Поле с номером (индексом) структуры информации о получателе CMS-сообщения (заполняется для типа структуры информации о получателе **FLAG_CMS_RECINF_TYPE_KARI**).*

– int **version**

Поле с номером версии структуры информации о получателе CMS-сообщения.

– cms_certid_t **originatorId**

Поле со структурой идентификатора сертификата отправителя CMS-сообщения (данное поле не заполняется для CMS-сообщений, зашифрованных анонимным способом).

– String **publicKeyAlg**

*Поле с строкой объектного идентификатора (OID) алгоритма открытого ключа сертификата получателя (заполняется для типа структуры информации о получателе **FLAG_CMS_RECINF_TYPE_KARI**):*

- **"1.2.643.7.1.1.1.1"** - открытый ключ по ГОСТ Р 34.10-2012 (256 бит);
- **"1.2.643.7.1.1.1.2"** - открытый ключ по ГОСТ Р 34.10-2012 (512 бит).

– String **cipherAlg**

Поле с строкой объектного идентификатора (OID) алгоритма согласования или зашифрования сеансового ключа:

- **"1.2.643.7.1.1.1.1"** - согласование ключа по ГОСТ Р 34.10-2012 (256 бит) (для CMS-сообщений, зашифрованных по ГОСТ 28147-89);
- **"1.2.643.7.1.1.1.2"** - согласование ключа по ГОСТ Р 34.10-2012 (512 бит) (для CMS-сообщений, зашифрованных по ГОСТ 28147-89);
- **"1.2.643.7.1.1.7.1.1"** - зашифрование (экспорт) сеансового ключа по ГОСТ Р 34.12-2015 (блочный шифр «Магма»);
- **"1.2.643.7.1.1.7.2.1"** - зашифрование (экспорт) сеансового ключа по ГОСТ Р 34.12-2015 (блочный шифр «Кузнечик»).

– cms_certid_t **recipientId**

Поле со структурой идентификатора сертификата получателя CMS-сообщения.

Структура cms_msginf_t

Структура информации о CMS-сообщении:

– int **type**

Поле с типом CMS-сообщения:

- **FLAG_CMS_MSGINF_TYPE_SIGN** - CMS-сообщение является подписанным (формат сообщения CMS Signed);
- **FLAG_CMS_MSGINF_TYPE_ENVL** - CMS-сообщение является зашифрованным (формат сообщения CMS Enveloped).

– int **flags**

Поле с маской (побитовым ИЛИ) флагов CMS-сообщения:

- **FLAG_CMS_MSGINF_FLAG_NDEF** - CMS-сообщение закодировано с помощью ASN.1 кодировки неопределенной длины (ASN.1 indefinite-length encoding). Если флаг не установлен, то используется ASN.1 кодировка определенной длины (ASN.1 definite-length encoding);
- **FLAG_CMS_MSGINF_FLAG_DTCH** - подписанное CMS-сообщение является сообщением с отделенными ЭП. Если флаг не установлен, то подписанное CMS-сообщение - с совмещенными ЭП (данный флаг актуален только для CMS-сообщений типа **FLAG_CMS_MSGINF_TYPE_SIGN**).

– int **signer_num**

Количество элементов массива структур с информацией о подписантах CMS-сообщения (заполняется только для CMS-сообщений типа **FLAG_CMS_MSGINF_TYPE_SIGN**).

– cms_siginf_t[] **signers**

Поле с массивом структур с информацией о подписантах CMS-сообщения (заполняется только для CMS-сообщений типа **FLAG_CMS_MSGINF_TYPE_SIGN**).

– int **recipient_num**

Количество элементов массива структур с информацией о получателях CMS-сообщения (заполняется только для CMS-сообщений типа **FLAG_CMS_MSGINF_TYPE_ENVL**).

– cms_recinf_t[] **recipients**

Поле с массивом структур с информацией о получателях CMS-сообщения (заполняется только для CMS-сообщений типа **FLAG_CMS_MSGINF_TYPE_ENVL**).

– String **cipherAlg**

Поле с строкой объектного идентификатора (OID) алгоритма шифрования данных (заполняется только для CMS-сообщений типа **FLAG_CMS_MSGINF_TYPE_ENVL**):

- **"1.2.643.2.2.21"** - шифрование по ГОСТ 28147-89 в режиме гаммирования с обратной связью;
- **"1.2.643.7.1.1.5.1.1"** - шифрование по ГОСТ Р 34.12-2015 (блочный шифр «Магма») в режиме гаммирования;
- **"1.2.643.7.1.1.5.1.2"** - шифрование по ГОСТ Р 34.12-2015 (блочный шифр «Магма») в режиме гаммирования с вычислением имитовставки;
- **"1.2.643.7.1.1.5.2.1"** - шифрование по ГОСТ Р 34.12-2015 (блочный шифр «Кузнечик») в режиме гаммирования;

• **"1.2.643.7.1.1.5.2.2"** - шифрование по ГОСТ Р 34.12-2015 (блочный шифр «Кузнечик») в режиме гаммирования с вычислением имитовставки.

– int **unprotAttr_num**

*Количество элементов массива строк объектных идентификаторов незащищенных атрибутов (заполняется только для CMS-сообщений типа **FLAG_CMS_MSGINF_TYPE_ENVL**).*

– String[] **unprotAttrs**

*Поле с массивом строк объектных идентификаторов незащищенных атрибутов (заполняется только для CMS-сообщений типа **FLAG_CMS_MSGINF_TYPE_ENVL**).*

1.24.2 Функции блочного получения информации о CMS-сообщениях

int **VCERT_CmsBlkMsgInfMem** (mem_blk_t cms, cms_msginf_t pMsgInf, int flag)

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

Аргументы:

– **cms** (in) блок памяти с CMS-сообщением (должно выполняться условие **cms->len > 0**);

– **pMsgInf** (out) структура с информацией о CMS-сообщении;

– **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsBlkMsgInfFile** (String cms, cms_msginf_t pMsgInf, int flag)

Функция блочного получения информации о CMS-сообщениях в виде файла

Аргументы:

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

– **pMsgInf** (out) структура с информацией о CMS-сообщении;

– **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.24.3 Функции потокового получения информации о CMS-сообщениях

int **VCERT_CmsStrMsgInfInitMem** (mem_blk_t cms, strcms_handle_t hStr, int flag)

Функция инициализации потокового получения информации о блоке памяти CMS

Аргументы:

– **cms** (in) блок памяти с началом CMS-сообщения (должно выполняться условие **cms->len ≥ 128**);

– **hStr** (out) контекст потоковой операции;

– **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrMsgInfUpdateMem** (strcms_handle_t hStr, mem_blk_t cms)

Функция продолжения потокового получения информации о блоке памяти CMS

Аргументы:

- **hStr** (in) контекст потоковой операции;
- **cms** (in) блок памяти с продолжением CMS-сообщения.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrMsgInfFinalMem** (strcms_handle_t hStr, cms_msginf_t pMsgInf)

Функция финализации потокового получения информации о блоке памяти CMS

Аргументы:

- **hStr** (in) контекст потоковой операции;
- **pMsgInf** (out) структура с информацией о CMS-сообщении.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_CmsStrMsgInfFile** (String cms, cms_msginf_t pMsgInf, int flag)

Функция потокового получения информации о CMS-сообщениях в виде файла

Аргументы:

- **cms** (in) файл (не нулевой длины) с CMS-сообщением;
- **pMsgInf** (out) структура с информацией о CMS-сообщении;
- **flag** (in) маска флагов (зарезервировано, должно быть равно 0).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.25 Простановка и проверка штампов времени

Подробное описание процессов простановки и проверки штампов времени ЭП CMS-сообщений приводится в документе ВАМБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

Следует учитывать тот факт, что при простановке штампа времени подписанного CMS-сообщения необходимо использовать те функции простановки, которые не изменяют тип ASN.1 кодировки - определенной или неопределенной длины.

1.25.1 Структуры простановки и проверки штампов времени

Структура tsp_request_param_t

Структура параметров простановки штампа времени CMS-сообщений:

– int **flag**

Поле с маской (побитовым ИЛИ) флагов простановки штампа времени CMS-сообщений:

- **FLAG_TSP_REQUEST_INCLUDENONCE** - добавлять в запрос на получение штампа времени индикатор включения в штамп случайного числа (**nonce**);

- **FLAG_TSP_REQUEST_ATTACHEDSIGNER** - добавлять в запрос на получение штампа времени индикатор включения в штамп сертификата сервера штампов времени.

– certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

– int **index**

Поле с порядковым номером (индексом) ЭП CMS-сообщения, для которой следует запросить штамп времени (должно выполняться условие **index** ≥ 0).

Структура tsp_response_param_t

Структура параметров вычисления ЭП и формирования штампов времени:

– int **flag**

Поле с маской (побитовым ИЛИ) флагов вычисления ЭП и формирования штампов времени:

- **FLAG_TSP_RESPONSE_INCLUDETSANAME** - добавлять в формируемый штамп времени X.500-имя владельца сертификата сервера штампов времени.

– certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

Структура tsp_verify_param_t

Структура параметров проверки штампа времени CMS-сообщений:

– int **flag**

Поле с маской (побитовым ИЛИ) флагов проверки штампа времени CMS-сообщений:

- **FLAG_TSP_VERIFY_IGNOREATTACHEDSIGNER** - не искать сертификаты подписантов среди сертификатов и САС, прикрепленных к штампу времени.

– certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

– int **index**

Поле с порядковым номером (индексом) ЭП CMS-сообщения, для которой следует проверить штамп времени (должно выполняться условие **index** ≥ 0).

– int **info**

Поле с требуемой возвращаемой информацией о сертификате сервера штампов времени.

Структура tsp_verify_result_t

Структура результата проверки штампа времени для заданной ЭП:

– long **timeEx**

Поле со временем простановки (вычисления ЭП) штампа времени в часовом поясе UTC.

– certificate_t **cert**

*Поле со структурой сертификата, на котором выполнялась проверка ЭП, т.е. сертификата сервера штампов времени (равно **NULL** в случае ненахождения сертификата сервера штампов времени).*

1.25.2 Функции блочной простановки и проверки штампов времени

int **VCERT_TspBlkRequestFromCmsMem** (tsp_request_param_t pRequestPara, mem_blk_t icms, mem_blk_t oreq)

Функция блочного создания запроса на получение штампа времени блока памяти

Аргументы:

– **pRequestPara** (in) структура параметров простановки штампа времени CMS-сообщений;

– **icms** (in) блок памяти с подписанным CMS-сообщением (должно выполняться условие **icms->len > 0**);

– **oreq** (out) блок памяти с запросом на получение штампа времени.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspBlkRequestFromCmsFile** (tsp_request_param_t pRequestPara, String icms, mem_blk_t oreq)

Функция блочного создания запроса на получение штампа времени файла

Аргументы:

– **pRequestPara** (in) структура параметров простановки штампа времени CMS-сообщений;

– **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением;

– **oreq** (out) блок памяти с запросом на получение штампа времени.

Возвращаемые значения:

– **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspSignResponse** (tsp_response_param_t pResponsePara, mem_blk_t ireq, mem_blk_t ores)

Функция блочного вычисления ЭП и формирования штампов времени

Аргументы:

– **pResponsePara** (in) структура параметров вычисления ЭП и формирования

штампов времени;

- **ireq** (in) блок памяти с запросом на получение штампа времени (должно выполняться условие **ireq->len > 0**);
- **ores** (out) блок памяти с сформированным штампом времени.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspVerifyResponse** (tsp_verify_param_t pVerifyPara, mem_blk_t ires, tsp_verify_result_t pVerifyResult)

Функция блочной проверки ЭП сформированного штампа времени

Аргументы:

- **pVerifyPara** (in) структура параметров проверки штампа времени CMS-сообщений;
- **ires** (in) блок памяти с сформированным штампом времени (должно выполняться условие **ires->len > 0**);
- **pVerifyResult** (out) структура результата проверки штампа времени для заданной ЭП.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspBlkUrlStampCmsMem** (tsp_request_param_t pRequestPara, String url, mem_blk_t icms, mem_blk_t ocms)

Функция блочной простановки штампа времени блока памяти CMS на сервере

Аргументы:

- **pRequestPara** (in) структура параметров простановки штампа времени CMS-сообщений;
- **url** (in) строка (URI) с адресом сервера штампов времени;
- **icms** (in) блок памяти с подписанным CMS-сообщением (должно выполняться условие **icms->len > 0**);
- **ocms** (out) блок памяти с подписанным CMS-сообщением, включающим штамп времени.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspBlkUrlStampCmsFile** (tsp_request_param_t pRequestPara, String url, String icms, String ocms)

Функция блочной простановки штампа времени файла CMS на сервере

Аргументы:

- **pRequestPara** (in) структура параметров простановки штампа времени CMS-сообщений;
- **url** (in) строка (URI) с адресом сервера штампов времени;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением;

– **ocms** (out) файл с подписанным CMS-сообщением, включающим штамп времени.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspBlkVerifyCmsMem** (tsp_verify_param_t pVerifyPara, mem_blk_t icms, tsp_verify_result_t pVerifyResult)

Функция блочной проверки штампа времени блока памяти CMS

Аргументы:

- **pVerifyPara** (in) структура параметров проверки штампа времени CMS-сообщений;
- **icms** (in) блок памяти с подписанным CMS-сообщением, включающим штамп времени (должно выполняться условие **icms->len > 0**);
- **pVerifyResult** (out) структура результата проверки штампа времени для заданной ЭП.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspBlkVerifyCmsFile** (tsp_verify_param_t pVerifyPara, String icms, tsp_verify_result_t pVerifyResult)

Функция блочной проверки штампа времени файла CMS

Аргументы:

- **pVerifyPara** (in) структура параметров проверки штампа времени CMS-сообщений;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением, включающим штамп времени;
- **pVerifyResult** (out) структура результата проверки штампа времени для заданной ЭП.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.25.3 Функции потоковой простановки и проверки штампов времени

int **VCERT_TspStrUrlStampCmsInitMem** (tsp_request_param_t pRequestPara, String url, mem_blk_t icms, strcms_handle_t hStr)

Функция инициализации потоковой простановки штампа времени CMS на сервере

Аргументы:

- **pRequestPara** (in) структура параметров простановки штампа времени CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **url** (in) строка (URI) с адресом сервера штампов времени (должна быть одинакова для функций инициализации, продолжения и финализации);

- **icms** (in) блок памяти с началом подписанного CMS-сообщения (должно выполняться условие **icms->len** \geq **128**);
- **hStr** (out) контекст потоковой операции.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspStrUrlStampCmsUpdateMem** (tsp_request_param_t pRequestPara, String url, strcms_handle_t hStr, mem_blk_t icms, mem_blk_t ocms)

Функция продолжения потоковой простановки штампа времени CMS на сервере

Аргументы:

- **pRequestPara** (in) структура параметров простановки штампа времени CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **url** (in) строка (URI) с адресом сервера штампов времени (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **icms** (in) блок памяти с продолжением подписанного CMS-сообщения;
- **ocms** (out) блок памяти с продолжением подписанного CMS-сообщения, включающего штамп времени.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspStrUrlStampCmsFinalMem** (tsp_request_param_t pRequestPara, String url, strcms_handle_t hStr, mem_blk_t ocms)

Функция финализации потоковой простановки штампа времени CMS на сервере

Аргументы:

- **pRequestPara** (in) структура параметров простановки штампа времени CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **url** (in) строка (URI) с адресом сервера штампов времени (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **ocms** (out) блок памяти с окончанием подписанного CMS-сообщения, включающего штамп времени.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspStrUrlStampCmsFile** (tsp_request_param_t pRequestPara, String url, String icms, String ocms)

Функция потоковой простановки штампа времени файла CMS на сервере

Аргументы:

- **pRequestPara** (in) структура параметров простановки штампа времени CMS-сообщений;
- **url** (in) строка (URI) с адресом сервера штампов времени;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением;
- **ocms** (out) файл с подписанным CMS-сообщением, включающим штамп времени.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspStrVerifyCmsInitMem** (tsp_verify_param_t pVerifyPara, mem_blk_t icms, strcms_handle_t hStr)

Функция инициализации потоковой проверки штампа времени блока памяти CMS

Аргументы:

- **pVerifyPara** (in) структура параметров проверки штампа времени CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **icms** (in) блок памяти с началом подписанного CMS-сообщения, включающего штамп времени (должно выполняться условие **icms->len** \geq 128);
- **hStr** (out) контекст потоковой операции.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspStrVerifyCmsUpdateMem** (tsp_verify_param_t pVerifyPara, strcms_handle_t hStr, mem_blk_t icms)

Функция продолжения потоковой проверки штампа времени блока памяти CMS

Аргументы:

- **pVerifyPara** (in) структура параметров проверки штампа времени CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);
- **hStr** (in) контекст потоковой операции;
- **icms** (in) блок памяти с продолжением подписанного CMS-сообщения, включающего штамп времени.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspStrVerifyCmsFinalMem** (tsp_verify_param_t pVerifyPara, strcms_handle_t hStr, tsp_verify_result_t pVerifyResult)

Функция финализации потоковой проверки штампа времени блока памяти CMS

Аргументы:

- **pVerifyPara** (in) структура параметров проверки штампа времени CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);

- **hStr** (in) контекст потоковой операции;
- **pVerifyResult** (out) структура результата проверки штампа времени для заданной ЭП.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TspStrVerifyCmsFile** (tsp_verify_param_t pVerifyPara, String icms, tsp_verify_result_t pVerifyResult)

Функция потоковой проверки штампа времени файла CMS

Аргументы:

- **pVerifyPara** (in) структура параметров проверки штампа времени CMS-сообщений;
- **icms** (in) файл (не нулевой длины) с подписанным CMS-сообщением, включающим штамп времени;
- **pVerifyResult** (out) структура результата проверки штампа времени для заданной ЭП.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.26 Получение online-статуса сертификата

Подробное описание процесса получения online-статуса сертификата приводится в документе ВАМБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

1.26.1 Структуры получения online-статуса сертификата

Структура ocsp_request_param_t

Структура параметров получения online-статуса сертификата:

- int **flag**

Поле с маской флагов (зарезервировано, должно быть равно 0).

- certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

Структура ocsp_response_param_t

Структура параметров вычисления ЭП online-статуса сертификата:

- int **flag**

Поле с маской флагов (зарезервировано, должно быть равно 0).

- certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

Структура ocsp_verify_param_t

Структура параметров проверки ЭП online-статуса сертификата:

– int **flag**

Поле с маской флагов (зарезервировано, должно быть равно **0**).

– certid_t **mycert**

Структура идентификатора сертификата для определения контекста выполнения (должен быть null при использовании локального Средства КЗИ).

– int **info**

Поле с требуемой возвращаемой информацией о сертификате сервера OCSP ответчика.

Структура ocsf_verify_result_t

Структура результата проверки ЭП online-статуса сертификата:

– int **status**

Поле с online-статусом сертификата:

- **0** - сертификат действителен;
- **1** - сертификат аннулирован;
- **2** - online-статус не известен.

– int **reason**

Для аннулированного сертификата, поле с причиной аннулирования сертификата (см. описание в подразделе 1.9).

– long **revtimeEx**

Для аннулированного сертификата, поле со временем аннулирования сертификата.

– long **thisupdEx**

Поле со временем начала действия данного online-статуса сертификата.

– long **nextupdEx**

Поле со временем окончания действия данного online-статуса сертификата.

– certificate_t **cert**

Поле со структурой сертификата, на котором выполнялась проверка ЭП, т.е. сертификата сервера OCSP ответчика (равно **NULL** в случае ненахождения сертификата сервера OCSP ответчика).

1.26.2 Функции получения online-статуса сертификата

int **VCERT_OcsfCreateRequest** (ocsf_request_param_t pRequestPara, mem_blk_t cert, certificate_t rqst)

Функция создания запроса на получение online-статуса сертификата

Аргументы:

– **pRequestPara** (in) структура параметров получения online-статуса сертификата;

– **cert** (in) блок памяти с сертификатом для получения online-статуса в DER-кодировке или PEM-формате;

– **rqst** (out) блок памяти с созданным запросом на получение online-статуса сертификата.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_OcspSignResponse** (ocsp_response_param_t pResponsePara, mem_blk_t rqst, mem_blk_t resp)

Функция обработки запроса на получение online-статуса сертификата

Аргументы:

- **pResponsePara** (in) структура параметров вычисления ЭП online-статуса сертификата;
- **rqst** (in) блок памяти с запросом на получение online-статуса сертификата (должно выполняться условие **rqst->len > 0**);
- **resp** (out) блок памяти с подписанным online-статусом сертификата.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_OcspUrlObtainResponse** (ocsp_request_param_t pRequestPara, String url, mem_blk_t cert, mem_blk_t resp)

Функция получения online-статуса сертификата на заданном OSCP сервере

Аргументы:

- **pRequestPara** (in) структура параметров получения online-статуса сертификата;
- **url** (in) строка (URI) с адресом сервера OSCP ответчика. Если данная строка равна **NULL**, то для получения online-статуса последовательно используются точки AIA типа **id-ad-ocsp** с объектным идентификатором (OID) 1.3.6.1.5.5.7.48.1 из сертификата для получения online-статуса;
- **cert** (in) блок памяти с сертификатом для получения online-статуса в DER-кодировке или PEM-формате;
- **resp** (out) блок памяти с подписанным online-статусом сертификата.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_OcspVerifyResponse** (ocsp_verify_param_t pVerifyPara, mem_blk_t resp, ocsp_verify_result_t pVerifyResult)

Функция проверки ЭП online-статуса сертификата

Аргументы:

- **pVerifyPara** (in) структура параметров проверки ЭП online-статуса сертификата;
- **resp** (in) блок памяти с подписанным online-статусом сертификата (должно выполняться условие **resp->len > 0**);
- **pVerifyResult** (out) структура результата проверки ЭП online-статуса сертификата.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.27 Протокол безопасности транспортного уровня TLS 1.2

Подробное описание процесса передачи данных посредством установления защищенного канала по протоколу Transport Layer Security приводится в документе ВАМБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

1.27.1 Функции протокола безопасности транспортного уровня TLS 1.2

int VCERT_TlsSessionCreate (String server, tls_handle_t hTls, int flag)

Функция создания контекста защищенного канала по протоколу TLS

Аргументы:

- **server** (in) 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;
- **hTls** (out) контекст защищенного канала по протоколу TLS;
- **flag** (in) маска (побитовое ИЛИ) флагов создания контекста защищенного канала по протоколу TLS (FLAG_TLS_XXX).

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int VCERT_TlsSessionDestroy (tls_handle_t hTls)

Функция освобождения контекста защищенного канала по протоколу TLS

Аргументы:

- **hTls** (in) контекст защищенного канала по протоколу TLS.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int VCERT_TlsSessionHandshake (tls_handle_t hTls, mem_blk_t itls, mem_blk_t otls)

Функция выполнения переговоров при создании защищенного канала TLS

Аргументы:

- **hTls** (in) контекст защищенного канала по протоколу TLS;
- **itls** (in) блок памяти с данными протокола TLS, полученными от противоположной стороны (должно выполняться условие **itls->len** \geq **0**). При формировании первого сообщения клиентом длина должна быть установлена в **0**;
- **otls** (out) блок памяти с данными протокола TLS, отправляемыми противоположной стороне.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int VCERT_TlsSessionComplete (tls_handle_t hTls)

Функция определения состояния защищенного канала по протоколу TLS**Аргументы:**

- **hTls** (in) контекст защищенного канала по протоколу TLS.

Возвращаемые значения:

- **VCERT_OK** защищенный канал между клиентом и сервером уже установлен;
- **VCERT_E_TLS_NOT_COMPLETE** 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 функцию **VCERT_TlsSessionHandshake()**;
 - при переговорах, проходящих уже после создания защищенного канала, необходимо вызвать функцию **VCERT_TlsSessionWrite()**, установив длину блока памяти с защищаемыми данными в **0**, и передать полученные данные TLS протокола противоположной стороне;
- **!= VCERT_OK && != VCERT_E_TLS_NOT_COMPLETE** в случае ошибки.

int **VCERT_TlsSessionQuery** (tls_handle_t hTls, int query, mem_blk_t data)

Функция получения данных или управления защищенным каналом TLS**Аргументы:**

- **hTls** (in) контекст защищенного канала по протоколу TLS;
- **query** (in) запрашиваемые данные или команда управления (VCERT_TLS_QRY_XXX);
- **data** (out) блок памяти с запрошенными данными или результатом команды управления.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TlsSessionWrite** (tls_handle_t hTls, mem_blk_t itls, mem_blk_t otls)

Функция отправки данных другой стороне защищенного канала TLS**Аргументы:**

- **hTls** (in) контекст защищенного канала по протоколу TLS;
- **itls** (in) блок памяти с защищаемыми данными, предназначенными для передачи противоположной стороне (должно выполняться условие **itls->len ≥ 0**);
- **otls** (out) блок памяти с данными протокола TLS, отправляемыми противоположной стороне.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

int **VCERT_TlsSessionRead** (tls_handle_t hTls, mem_blk_t itls, mem_blk_t otls)

Функция получения данных от другой стороны защищенного канала TLS**Аргументы:**

- **hTls** (in) контекст защищенного канала по протоколу TLS;
- **itls** (in) блок памяти с данными протокола TLS, полученными от противоположной стороны (должно выполняться условие **itls->len** \geq 0);
- **otls** (out) блок памяти с защищаемыми данными, переданными противоположной стороной.

Возвращаемые значения:

- **VCERT_OK** защищаемые данные были успешно получены от противоположной стороны;
- **VCERT_E_TLS_READ_MORE** необходимо продолжить чтение данных от противоположной стороны. Перед чтением следует вызовом функции **VCERT_TlsSessionComplete()** определить состояние защищенного канала и, при получении кода ошибки **VCERT_E_TLS_NOT_COMPLETE**, продолжить переговоры;
- **!= VCERT_OK && != VCERT_E_TLS_READ_MORE** в случае ошибки.

1.28 Преобразование в формат и из формата Base64

Подробное описание Функций преобразования бинарных данных в формат и из формата Base64 приводится в документе ВАНБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

1.28.1 Функции преобразования в формат и из формата Base64

int VCERT1API **VCERT_EncodeMem** (byte[] t, byte[] f, int n)

Функция закодирования бинарных данных в формат Base64

Аргументы:

- **t** (out) заранее выделенный буфер для размещения закодированных данных. Размер данного буфера должен быть не менее чем **VCERT_BASE64_LEN(n)** байтов;
- **f** (in) исходный буфер с бинарными данными для закодирования;
- **n** (in) размер исходного буфера с бинарными данными в байтах (должен быть в интервале от 1 до $2^{30} - 1$).

Возвращаемые значения:

- **реальный размер закодированных данных** в случае успеха или отрицательное значение (\geq 0x80000000) в случае ошибки.

int VCERT1API **VCERT_DecodeMem** (byte[] t, const byte[] f, int n)

Функция раскодирования бинарных данных из формата Base64

Аргументы:

- **t** (out) заранее выделенный буфер для размещения раскодированных данных. Размер данного буфера должен быть не менее чем **n** байтов;
- **f** (in) исходный буфер с закодированными данными в кодировке Base64;
- **n** (in) размер исходного буфера с закодированными данными в байтах (должен быть в интервале от 1 до $2^{31} - 1$).

Возвращаемые значения:

– **реальный размер раскодированных данных** в случае успеха или отрицательное значение ($\geq 0x80000000$) в случае ошибки.

1.29 Выработка случайного числа заданной длины

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

1.29.1 Функции выработки случайного числа заданной длины

int VCERT_GenRandom (int size, mem_blk_t rand)

Функция выработки случайного числа заданной длины

Аргументы:

- **size** (in) длина случайного числа для выработки в байтах;
- **rand** (out) блок памяти для записи случайного числа.

Возвращаемые значения:

- **VCERT_OK** в случае успеха или ненулевой код ошибки.

1.30 Описание конфигурационного файла pki1.conf, XML-шаблона и XML-запроса

Подробное описание конфигурационного файла pki1.conf, XML-шаблона и XML-запроса приводится в документе ВАМБ.00106-06 33 01 «СКАД «Сигнатура» версия 6. «Сигнатура-клиент» версия 6. Руководство программиста».

2 ОПИСАНИЕ ОШИБОЧНЫХ СИТУАЦИЙ

Ниже (Таблица 1) приведено описание возможных ошибочных ситуаций. В левой колонке указано символьное имя ошибки и шестнадцатеричное значение ее кода, в правой колонке приведено детальное описание и причина возникновения ошибки.

Таблица 1 – Описание ошибочных ситуаций

Имя и код ошибки	Описание и причина возникновения ошибки
VCERT_OK (0x00000000)	Успешное завершение функции
VCERT_E_GENERIC (0xE0700001)	Общая (внутренняя) ошибка библиотеки. Указывает на возможную ошибку в самой библиотеке или на искажения в ее настройках
VCERT_E_INVALID_PARAMETER (0xE0700002)	В функцию был передан неверный параметр. Возникает в случае передачи нулевого указателя, неверно заполненной структуры объекта системы управления сертификатами (СУС) или параметров или при неверном размере блока памяти
VCERT_E_INVALID_CONTEXT (0xE0700003)	Неверный контекст библиотеки, потоковой или другой операции. Вероятно, искажены настройки профиля пользователя или обнаружена ошибка в синтаксисе конфигурационного файла pkil.conf
VCERT_E_OPERATION_NOT_SUPPORTED (0xE0700004)	Операция (или функция) не поддерживается. Выполнен вызов функции или операции, не поддерживаемой библиотекой или не разрешенной для контекста библиотеки
VCERT_E_INVALID_FLAG (0xE0700005)	В функцию был передан неверный флаг. В параметре или в структуре параметров функции указана неверная маска (побитовое ИЛИ) флагов
VCERT_E_NO_MEMORY (0xE0700006)	Недостаточно оперативной памяти. Вероятно, произведен вызов блочной функции над слишком большим блоком памяти или файлом
VCERT_E_DIGEST (0xE0700007)	Ошибка вычисления хэш-значения. Вероятно, неверен объектный идентификатор (OID) алгоритма хэширования
VCERT_E_CERT_USAGE (0xE0700008)	Неверное использование сертификата. В рабочем сертификате отсутствует требуемое разрешенное использование ключа проверки ЭП/открытого ключа шифрования, регламент или расширенное использование ключа проверки ЭП/открытого ключа шифрования
VCERT_E_CERT_FIND_PRIVATE_KEY (0xE0700009)	Не найден ключ ЭП, соответствующий данному сертификату. Отсутствует ключевой носитель с требуемым ключом ЭП, неверен ПИН-код устройства типа смарт-карта или неверен пароль ключа ЭП
VCERT_E_CMS_ADD_SIGNATURE (0xE070000C)	Ошибка добавления ЭП к сообщению в формате CMS/PKCS#7. Вероятно, что недостаточно ресурсов для выполнения операции, произошел сбой аппаратного датчика случайных чисел (ДСЧ) или нет доступа к ФКН vdToken с неизвлекаемым ключом ЭП
VCERT_E_CMS_ASN1_DECODE (0xE070000F)	Ошибка выполнения ASN.1-распаковки сообщения в формате CMS/PKCS#7. Вероятно, CMS-сообщение повреждено или искажено
VCERT_E_CMS_ASN1_ENCODE (0xE0700010)	Ошибка выполнения ASN.1-упаковки сообщения в формате CMS/PKCS#7. Вероятно, возникла нехватка ресурсов для выполнения операции
VCERT_E_SIGN_HASH (0xE0700012)	Ошибка вычисления ЭП хэш-значения. Вероятно, неверна длина хэш-значения, произошел сбой аппаратного ДСЧ или нет доступа к ФКН vdToken с неизвлекаемым ключом ЭП
VCERT_E_VERIFY_POLICY (0xE0700013)	Ошибка добавления регламента в контекст проверки сертификата. Вероятно, объектный идентификатор (OID) регламента неверен
VCERT_E_VERIFY_EXTKEYUSAGE (0xE0700014)	Ошибка добавления расширенного использования ключа в контекст проверки сертификата. Вероятно, объектный идентификатор (OID) расширенного использования ключа проверки ЭП/открытого ключа шифрования неверен
VCERT_E_OVERFLOW (0xE0700016)	Ошибка переполнения - либо данные слишком велики, либо буфер слишком мал. Вероятно, произведен вызов блочной функции над слишком большим блоком памяти или файлом
VCERT_E_PKCS10_DAMAGED (0xE0700017)	PKCS#10 запрос на сертификат поврежден или искажен
VCERT_E_REVREQ_DAMAGED (0xE0700018)	Запрос на аннулирование сертификата поврежден или искажен
VCERT_E_VERIFY (0xE0700019)	Общая ошибка проверки ЭП CMS/PKCS#7 сообщения. Возникла ошибка при проверке хотя бы одной ЭП CMS-сообщения
VCERT_E_CMS_INVALID_TYPE (0xE0700022)	Неверный тип содержимого сообщения в формате CMS/PKCS#7. Вероятно, в функцию проверки ЭП передано зашифрованное CMS-сообщение или наоборот
VCERT_E_CMS_NO_RECIPIENTS (0xE0700024)	Отсутствуют или неверны данные сертификатов получателей зашифрованного сообщения в формате CMS/PKCS#7. CMS-сообщение повреждено или искажено

ВАМБ.00115-06 33 01

Имя и код ошибки	Описание и причина возникновения ошибки
VCERT_E_CMS_NOT_RECIPIENT (0xE0700026)	Владелец сертификата не является получателем зашифрованного сообщения в формате CMS/PKCS#7. Идентификатор рабочего сертификата отсутствует в списке получателей зашифрованного CMS-сообщения
VCERT_E_CMS_KEY_DECRYPT (0xE0700027)	Ошибка расшифрования сеансового ключа зашифрованного CMS/PKCS#7 сообщения. Вероятно, CMS-сообщение повреждено или искажено, или нет доступа к ФКН vdToken с неизвлекаемым закрытым ключом шифрования
VCERT_E_DATA_DECRYPT (0xE0700028)	Ошибка расшифрования блока данных. Вероятно, CMS-сообщение повреждено или искажено
VCERT_E_RANDOM (0xE0700029)	Ошибка генерации случайного числа. Вероятно, произошел сбой аппаратного ДСЧ
VCERT_E_OPEN_CONFIG (0xE071002A)	Ошибка доступа к конфигурационному файлу pkil.conf . В текущем рабочем каталоге процесса не найден конфигурационный файл pkil.conf
VCERT_E_READ_CONFIG (0xE071002B)	Ошибка разбора конфигурационного файла pkil.conf . Обнаружена ошибка в формате конфигурационного файла pkil.conf
VCERT_E_NO_DEFAULT_CONFIG (0xE071002C)	Профиль по умолчанию не указан в конфигурационном файле pkil.conf . Вероятно, была произведена попытка инициализации контекста библиотеки с профилем по умолчанию
VCERT_E_OPEN_PSTORE (0xE070002D)	Ошибка доступа к ПСП или к подписанному справочнику. Вероятно, путь (URI) к ПСП или подписанному справочнику неверен
VCERT_E_OPEN_LOCALSTORE (0xE070002E)	Ошибка доступа к ЛСП. Вероятно, путь (URI) к ЛСП неверен
VCERT_E_VERIFY_STORE_USAGE (0xE070002F)	Подписанный справочник имеет неверный идентификатор использования. Вероятно, произведена попытка использовать подписанное обновление от Центра сертификации (ЦС) или Центра регистрации (ЦР) вместо ПСП или наоборот
VCERT_E_VERIFY_STORE (0xE0700030)	Ошибка проверки целостности ПСП или подписанного справочника. Вероятно, произошла ошибка построения или проверки цепочки сертификата подписанта или подписанный справочник поврежден или искажен
VCERT_E_OPEN_LDAPSTORE (0xE0700031)	Ошибка доступа к ССС. Вероятно, путь (URI) к ССС неверен, отсутствует сетевое подключение к ССС или доступ к ССС запрещен из-за отсутствия билета Kerberos
VCERT_E_VERIFY_CERT (0xE0700034)	Ошибка построения и проверки цепочки сертификата. Вероятно, срок действия рабочего сертификата или ключа ЭП истек, не найдены сертификат ЦС или САС, необходимые для построения цепочки, или срок действия САС истек
VCERT_E_CERT_MISSING (0xE0700035)	Сертификат издателя не был найден в доступных справочниках. В доступных справочниках отсутствует сертификат ЦС, необходимый для построения цепочки, при этом не разрешен или отсутствует доступ к точкам AIA
VCERT_E_CERT_EXPIRED (0xE0700036)	Срок действия сертификата уже истек
VCERT_E_CERT_DAMAGED (0xE0700037)	Сертификат поврежден или искажен
VCERT_E_CERT_BROKEN - CONSTRAINT (0xE0700038)	Нарушены базовые ограничения цепочки сертификата
VCERT_E_CERT_REVOKED (0xE0700039)	Сертификат был аннулирован издателем
VCERT_E_CERT_UNTRUSTED (0xE070003A)	Цепочка сертификации не оканчивается доверенным сертификатом. В ПСП отсутствует необходимый сертификат корневого ЦС
VCERT_E_CRL_MISSING (0xE070003B)	САС издателя не был найден в доступных справочниках. В доступных справочниках отсутствует САС, необходимый для построения цепочки, при этом не разрешен или отсутствует доступ к точкам CDP
VCERT_E_CRL_EXPIRED (0xE070003C)	Срок действия САС уже истек
VCERT_E_CRL_DAMAGED (0xE070003D)	САС поврежден или искажен
VCERT_E_CERT_BROKEN - HIERARCHY (0xE070003E)	Нарушено ограничение иерархии цепочки сертификата
VCERT_E_CHAIN_ERROR (0xE070003F)	Общая ошибка построения и проверки цепочки сертификата. Вероятно, цепочка слишком длинная
VCERT_E_INVALID_USAGE (0xE0700041)	Ошибка использования сертификата не по назначению. В проверяемом сертификате отсутствует требуемое разрешенное использование ключа проверки ЭП/открытого ключа шифрования, регламент или расширенное использование ключа проверки ЭП/открытого ключа шифрования

ВАМБ.00115-06 33 01

Имя и код ошибки	Описание и причина возникновения ошибки
VCERT_E_INVALID_SIGNATURE (0xE0700042)	ЭП недостоверна. Проверяемые данные повреждены или искажены, или неверен ключ проверки ЭП, который был использован для проверки ЭП
VCERT_E_PUBKEY_NOT_FOUND (0xE0700043)	У сертификата неизвестный ключ проверки ЭП/открытый ключ шифрования. Вероятно, неверен алгоритм ключа проверки ЭП/открытого ключа шифрования сертификата
VCERT_E_UPDATECRL (0xE0700045)	Общая ошибка обновления САС. Вероятно, при критичном обновлении одного из САС, находящихся в ЛСП, произошла ошибка
VCERT_E_CERT_NOT_FOUND (0xE0700046)	Сертификат не был найден в доступных справочниках. При поиске в доступных справочниках не был найден ни один сертификат, удовлетворяющий заданному шаблону
VCERT_E_CERT_NOT_YET_VALID (0xE0700047)	Срок действия сертификата еще не наступил
VCERT_E_NO_ATTACHED_SIGNER (0xE070004A)	Сертификат подписанта отсутствует в сообщении в формате CMS/PKCS#7. Вероятно, был установлен флаг проверки ЭП FLAG_CMS_VERIFY_REQUIREATTACHEDSIGNER
VCERT_E_KERBEROS_FAILURE (0xE070004B)	Ошибка получения или обновления билета Kerberos. Вероятно, нет доступа к Центру распределения ключей (Key Distribution Center, KDC) или имя пользователя и пароль неверны
VCERT_E_KEY_EXPIRED (0xE070004C)	Ключ ЭП/закрытый ключ шифрования уже истек
VCERT_E_KEY_NOT_YET_VALID (0xE070004D)	Ключ ЭП/закрытый ключ шифрования еще недействителен
VCERT_E_CRL_NOT_YET_VALID (0xE070004E)	Срок действия САС еще не наступил
VCERT_E_INIT_CSP (0xE070004F)	Ошибка выполнения инициализации Средства КЗИ. Вероятно, Средство КЗИ не установлено, или его конфигурация искажена
VCERT_E_ENUM_OBJECTS (0xE0700050)	Ошибка доступа к справочнику при переборе объектов. Вероятно, путь (URI) к справочнику неверен, или отсутствует подключение к справочнику по сети
VCERT_E_ENUM_NO_MORE (0xE0700051)	В справочнике больше нет объектов для перебора. Перебор справочника завершен, все объекты были успешно считаны
VCERT_E_INVALID_X500_NAME (0xE0700052)	Текстовая строка, содержащая X.500-имя, имеет неверное представление. Вероятно, строка с X.500-именем искажена или содержит неверный RDN
VCERT_E_INVALID_HEX_STRING (0xE0700053)	Текстовая строка, содержащая шестнадцатеричное число, имеет неверное представление. Текстовая строка с шестнадцатеричным числом должна иметь вид 00:01:0E:0F
VCERT_E_CMS_STREAM_MISMATCH (0xE0700054)	Обнаружено несоответствие между потоковым признаком обрабатываемых данных и вызванной функцией. Вероятно, произошла попытка вызова блочной функции для обработки CMS-сообщения, имеющего ASN.1 кодировку неопределенной длины, или наоборот
VCERT_E_CMS_DETACH_MISMATCH (0xE0700055)	Обнаружено несоответствие между признаком отсоединенной ЭП обрабатываемых данных и вызванной функцией. Вероятно, произошла попытка вызова функции, предназначенной для обработки CMS-сообщений с присоединенными ЭП, для обработки CMS-сообщения с отсоединенными ЭП, или наоборот
VCERT_E_CMS_INVALID_DIGESTS (0xE0700056)	Отсутствуют или неверны данные алгоритмов хэширования подписанного сообщения в формате CMS/PKCS#7. Вероятно, CMS-сообщение повреждено или искажено
VCERT_E_CMS_INVALID_SIGNERS (0xE0700057)	Отсутствуют или неверны данные сертификатов подписантов подписанного сообщения в формате CMS/PKCS#7. Вероятно, CMS-сообщение повреждено или искажено
VCERT_E_CMS_INVALID_CIPHER (0xE0700058)	Зашифрованное сообщение в формате CMS/PKCS#7 содержит неизвестный или неверный алгоритм шифрования. Вероятно, CMS-сообщение повреждено или искажено
VCERT_E_CMS_DATA_SIGNING (0xE0700059)	Ошибка вычисления ЭП подписанного сообщения в формате CMS/PKCS#7. Вероятно, что недостаточно ресурсов для выполнения операции, произошел сбой аппаратного ДСЧ или нет доступа к ФКН vdToken с неизвлекаемым ключом ЭП
VCERT_E_CMS_OMAC_MISMATCH (0xE070005A)	Имитовставка зашифрованного сообщения в формате CMS/PKCS#7 не совпадает с вычисленной. Вероятно, CMS-сообщение повреждено или искажено
VCERT_E_FIND_SESSION (0xE0700081)	Требуемая сессия криптосервера не была найдена. В функцию библиотеки был передан идентификатор несуществующей сессии КС
VCERT_E_CMS_NOT_ENCRYPTED (0xE0700083)	CMS/PKCS#7-сообщение не зашифровано или формат сообщения поврежден или искажен. Вероятно, CMS-сообщение повреждено или искажено
VCERT_E_ADD_OBJECT (0xE0700087)	Ошибка добавления объекта в справочник сертификатов. Вероятно, такой объект уже существует или добавление объекта в ССС запрещено

ВАМБ.00115-06 33 01

Имя и код ошибки	Описание и причина возникновения ошибки
VCERT_E_TOO_MANY_CERTS_FOUND (0xE0720089)	Слишком много сертификатов найдено по уникальному критерию поиска. Вероятно, несколько сертификатов содержат один и тот же идентификатор ключа ЭП
VCERT_E_USER_CANCEL (0xE072008A)	Операция была отменена пользователем
VCERT_E_OPEN_INFILE (0xE070008B)	Ошибка открытия входного файла. Вероятно, путь или имя файла неверны или доступ к файлу запрещен
VCERT_E_OPEN_OUTFILE (0xE070008C)	Ошибка открытия выходного файла. Вероятно, путь или имя файла неверны или доступ к файлу запрещен
VCERT_E_READ_FILE (0xE070008D)	Ошибка чтения из входного файла. Вероятно, произошло искажение файловой системы
VCERT_E_WRITE_FILE (0xE070008E)	Ошибка записи в выходной файл. Вероятно, на файловой системе закончилось свободное пространство
VCERT_E_FILE_LENGTH (0xE070008F)	Неверный размер файла (нулевой или более 2Гб)
VCERT_E_DELETE_OBJECT (0xE0700091)	Ошибка удаления объекта из справочника сертификатов. Указанный объект не был удален из кэша контекста библиотеки или сессии КС или из ЛСП сессии КС по команде с АРМ УКС
VCERT_E_TOO_FEW_SIGNATURES (0xE0700092)	Подписанный документ содержит недостаточное количество ЭП. Вероятно, были установлены флаги проверки ЭП FLAG_CMS_VERIFY_DELETESIGNATURES или FLAG_CMS_VERIFY_MINIMUMSIGNATURES , или индекс ЭП для операции со штампом времени слишком велик
VCERT_E_GET_PUBKEY (0xE0700094)	Ошибка получения ключа проверки ЭП/открытого ключа шифрования сертификата. Вероятно, возникла нехватка ресурсов или неверен алгоритм ключа проверки ЭП/открытого ключа шифрования сертификата
VCERT_E_PKCS10_CREATE (0xE0700098)	Ошибка создания нового PKCS#10 запроса. Вероятно, произошла ошибка при генерации или записи ключа ЭП на ключевой носитель или XML шаблон имеет неверный формат
VCERT_E_PKCS10_SIGN (0xE070009A)	Ошибка вычисления ЭП PKCS#10 запроса. Вероятно, произошел сбой аппаратного ДСЧ или нет доступа к ФКН vdToken с неизвлекаемым ключом ЭП
VCERT_E_REVREQ_CREATE (0xE070009B)	Ошибка создания нового запроса на аннулирование. Вероятно, возникла нехватка ресурсов
VCERT_E_REVREQ_SIGN (0xE070009C)	Ошибка вычисления ЭП запроса на аннулирование. Вероятно, произошел сбой аппаратного ДСЧ или нет доступа к ФКН vdToken с неизвлекаемым ключом ЭП
VCERT_E_LOAD_PRIVATE_KEY (0xE070009D)	Ошибка загрузки ключа ЭП/закрытого ключа шифрования. Отсутствует ключевой носитель с требуемым ключом ЭП/закрытым ключом шифрования, неверен ПИН-код устройства типа смарт-карта или неверен пароль ключа ЭП/закрытого ключа шифрования
VCERT_E_ADD_SIGNER (0xE070009E)	Ошибка добавления ЭП к ЛСП или к подписанному справочнику. Вероятно, произошел сбой аппаратного ДСЧ или нет доступа к ФКН vdToken с неизвлекаемым ключом ЭП
VCERT_E_OPEN_IDP (0xE07000A1)	Ошибка доступа к точке распространения САС. Вероятно, к точке CDP запрещен доступ или в точке CDP отсутствует требуемый САС
VCERT_E_READ_IDP (0xE07000A2)	Ошибка чтения из точки распространения САС. Вероятно, возникла проблема с сетевым подключением или в точке CDP отсутствует требуемый САС
VCERT_E_INVALID_CREDENTIALS (0xE02000A8)	Ошибочные данные аутентификации при доступе к сессии криптосервера. Вероятно, длина данных аутентификации равна 0
VCERT_E_ACCESS_DENIED (0xE02000A9)	Доступ к сессии криптосервера запрещен. Вероятно, данные аутентификации неверны
VCERT_E_SESSION_BLOCKED (0xA02000AA)	Сессия криптосервера заблокирована
VCERT_E_CLIENT_INFO (0xE02000AB)	Ошибка получения информации о клиенте из протокола DCE-RPC. Вероятно, произошла системная ошибка библиотеки DCE-RPC при получении сетевого адреса клиента
VCERT_E_UNSECURE_CREDENTIALS (0xE02000AC)	Небезопасные (слишком короткие) данные аутентификации сессии криптосервера. Длина данных аутентификации должна быть не менее 8 символов
VCERT_E_SESSION_TIMEOUT (0xA07000AE)	Истек интервал ожидания доступа к сессии КС из-за того, что данная сессия КС в настоящий момент заблокирована и не готова обрабатывать поступающие запросы. Данная ошибка может возникнуть, только если для данной сессии КС настроен ненулевой интервал ожидания доступа

Имя и код ошибки	Описание и причина возникновения ошибки
VCERT_E_TSP_HASH_LENGTH (0xE0700100)	Неверная длина хэш-значения при создании запроса на штамп времени. Длина хэш-значения не соответствует указанному алгоритму хэширования
VCERT_E_TSP_HASH_ALGORITHM (0xE0700101)	Неверный алгоритм хэширования при создании запроса на штамп времени. Объектный идентификатор (OID) алгоритма хэширования неверен
VCERT_E_TSP_CERT_PURPOSE (0xE0700102)	Сертификат не может быть использован для подписи штампов времени. Сертификат не удовлетворяет условиям использования на сервере штампов времени
VCERT_E_TSP_SIGN_FAILED (0xE0700103)	Ошибка вычисления ЭП штампа времени. Вероятно, произошел сбой аппаратного ДСЧ или нет доступа к ФКН vdToken с неизвлекаемым ключом ЭП
VCERT_E_TSP_NO_DIGEST (0xE0700104)	В списке атрибутов отсутствует хэш-значение ЭП и/или данных. Вероятно, штамп времени поврежден или искажен
VCERT_E_TSP_INVALID_SIGNER_NUM (0xE0700105)	Штамп времени содержит неверное количество ЭП. Вероятно, штамп времени поврежден или искажен
VCERT_E_TSP_NO_TST_INFO (0xE0700106)	Ошибка при получении информационного блока штампа времени. Вероятно, штамп времени поврежден или искажен
VCERT_E_TSP_RESP_ASN1_DECODE (0xE0700107)	Ошибка выполнения ASN.1-распаковки подписанного штампа времени. Вероятно, штамп времени поврежден или искажен
VCERT_E_TSP_RESP_NOT_ISSUED (0xE0700108)	Штамп времени не был выдан авторитетным источником. Вероятно, произошла внутренняя ошибка сервера штампов времени
VCERT_E_TSP_DIGEST_MISMATCH (0xE0700109)	Штамп времени содержит хэш-значение, отличное от хэш-значения ЭП CMS/PKCS#7 сообщения. Вероятно, штамп времени поврежден или искажен
VCERT_E_OCSP_CERT_PURPOSE (0xE0700140)	Сертификат не может быть использован для вычисления ЭП ответов сетевого ответчика. Сертификат не удовлетворяет условиям использования на сервере OCSP ответчика
VCERT_E_OCSP_SIGN_FAILED (0xE0700141)	Ошибка вычисления ЭП ответа сетевого ответчика. Вероятно, произошел сбой аппаратного ДСЧ или нет доступа к ФКН vdToken с неизвлекаемым ключом ЭП
VCERT_E_OCSP_RESP_ASN1_DECODE (0xE0700142)	Ошибка выполнения ASN.1-распаковки подписанного ответа сетевого ответчика. Вероятно, ответ сервера OCSP ответчика поврежден или искажен
VCERT_E_OCSP_RESP_NOT_ISSUED (0xE0700143)	Подписанный ответ не был выдан сетевым ответчиком. Вероятно, произошла внутренняя ошибка сервера OCSP ответчика
VCERT_E_OCSP_NOT_BASICRESP (0xE0700144)	Неверный (небазовый) тип подписанного ответа сетевого ответчика. Вероятно, ответ сервера OCSP ответчика содержит статус более чем для одного сертификата
VCERT_E_OCSP_CERTID_MISMATCH (0xE0700145)	Идентификатор сертификата из подписанного ответа сетевого ответчика не соответствует запрашиваемому. Вероятно, ответ сервера OCSP ответчика поврежден или искажен
VCERT_E_OCSP_ISSUER_MISMATCH (0xE0700146)	Издатель сертификата сетевого ответчика не соответствует издателю проверяемого сертификата. Вероятно, ответ сервера OCSP ответчика поврежден или искажен
VCERT_E_TLS_UNSUPPORTED (0xE0700180)	Функции протокола TLS не могут быть использованы с данным контекстом библиотеки. Вероятно, произведена попытка использования функций протокола TLS с минимальным контекстом библиотеки
VCERT_E_TLS_NEW_CONTEXT (0xE0700181)	Ошибка создания контекста нового сеанса связи протокола TLS. Вероятно, возникла нехватка ресурсов, произошел сбой аппаратного ДСЧ или использован контекст проверки библиотеки
VCERT_E_TLS_INVALID_STATE (0xE0700182)	Контекст сеанса связи протокола TLS находится в неверном состоянии. Вероятно, произведена попытка обмена данными между клиентом и сервером, когда защищенный канал еще не сформирован
VCERT_E_TLS_HANDSHAKE (0xE0700183)	Ошибка выполнения переговоров при формировании нового сеанса связи протокола TLS. Клиент и сервер не смогли сформировать защищенный канал, вероятно из-за отсутствия общих наборов криптографических алгоритмов
VCERT_E_TLS_NOT_COMPLETE (0xE0700184)	Переговоры при формировании нового сеанса связи протокола TLS еще не завершены
VCERT_E_TLS_NO_QUERY_DATA (0xE0700185)	Запрашиваемые данные в контексте сеанса связи протокола TLS отсутствуют. Вероятно, на сервере был выполнен запрос на получение сертификата клиента в DER-кодировке при выполнении односторонней аутентификации
VCERT_E_TLS_WRONG_CERT (0xE0700186)	Сертификат противоположной стороны сеанса связи TLS протокола неверен или искажен
VCERT_E_TLS_WRONG_NAME (0xE0700187)	Сертификат противоположной стороны сеанса связи TLS протокола имеет неверное имя. Вероятно, сертификат сервера имеет в дополнении "Альтернативное имя владельца" DNS-имя отличное от того, которое указал клиент

ВАНБ.00115-06 33 01

Имя и код ошибки	Описание и причина возникновения ошибки
VCERT_E_TLS_WRITE_ERROR (0xE0700188)	Ошибка при записи данных сеанса связи протокола TLS. Вероятно, возникла нехватка ресурсов или данные TLS протокола искажены
VCERT_E_TLS_READ_ERROR (0xE0700189)	Ошибка при чтении данных сеанса связи протокола TLS. Вероятно, данные TLS протокола искажены
VCERT_E_TLS_READ_MORE (0xE0700190)	Следует продолжить чтение данных сеанса связи протокола TLS. Вероятно, необходимо продолжить переговоры для завершения создания защищенного канала
ERR_PROFILES_BAD_PARAM (0xE0D50001)	При вызове какой-либо функции библиотеки ей передан параметр с недопустимым значением - скорее всего нулевой указатель
ERR_PROFILES_BUFFER_SIZE (0xE0D50002)	При работе со строками (копирование, чтение из реестра и пр.) размер выделенного буфера недостаточен для размещения строки
ERR_PROFILES_NO_MEMORY (0xE0D50003)	Ошибка выделения памяти - либо произошло исчерпание памяти системы, либо при выделении памяти запрошен неадекватный размер
ERR_PROFILES_GET_INSTANCE (0xE0D50004)	Не инициализирована переменная CRYPTO_hinstance, содержащая HINSTANCE исполняемого модуля, содержащего ресурсы
ERR_PROFILES_CREATE_DLG (0xE0D50005)	Ошибка инициализации модального диалога - скорее всего испорчены ресурсы или неправильно инициализирована переменная CRYPTO_hinstance, содержащая HINSTANCE исполняемого модуля, содержащего ресурсы
ERR_PROFILES_GET_DLG_ITEM (0xE0D50006)	Ошибка доступа к элементам управления (кнопка, поле редактирования, список и т.д.) модального диалога - скорее всего испорчены ресурсы
ERR_PROFILES_GET_USER_DIR (0xE0D50007)	Ошибка при вызове функции SHGetFolderPath() библиотеки shell32.dll, возвращающей каталог пользователя по умолчанию - скорее всего проблемы с файловой системой
ERR_PROFILES_GET_WND_RECT (0xE0D50008)	Ошибка функции GetWindowRect() получающей координаты окна. Глобальные проблемы системы
ERR_PROFILES_SET_WND_POS (0xE0D50009)	Ошибка функции SetWindowPos() устанавливающей положение окна. Глобальные проблемы системы
ERR_PROFILES_CLN_TO_SCR (0xE0D5000A)	Ошибка функции ScreenToClient() приводящей экранные координаты окна к клиентским. Глобальные проблемы системы
ERR_PROFILES_USER_CANCEL (0xE0D5000B)	Пользователь нажал кнопку "Отмена" или клавишу ESC
ERR_PROFILES_NO_REG_KEY (0xE0D5000C)	Отсутствует ключ реестра
ERR_PROFILES_DONT_OPEN_- REG_KEY (0xE0D5000D)	Ошибка открытия ключа реестра
ERR_PROFILES_DONT_CREATE_- REG_KEY (0xE0D5000E)	Ошибка создания ключа реестра
ERR_PROFILES_ACCESS_DENY_- REG_KEY (0xE0D5000F)	Недостаточно прав для создания ключа в реестре
ERR_PROFILES_DONT_DEL_- REG_KEY (0xE0D50010)	Ошибка удаления ключа реестра
ERR_PROFILES_AC_DENY_DEL_- REG_KEY (0xE0D50011)	Недостаточно прав для удаления ключа в реестре
ERR_PROFILES_NO_REG_VAL (0xE0D50012)	Отсутствует значение в реестре
ERR_PROFILES_DONT_READ_- REG_VAL (0xE0D50013)	Ошибка чтения значения в реестре
ERR_PROFILES_DONT_WRITE_- REG_VAL (0xE0D50014)	Ошибка записи значения в реестр
ERR_PROFILES_ACCESS_DENY_- REG_VAL (0xE0D50015)	Недостаточно прав для записи значения в реестр
ERR_PROFILES_BAD_TYPE_- REG_VAL (0xE0D50016)	Неправильный тип значения в реестре
ERR_PROFILES_DONT_ENUM_- REG_VAL (0xE0D50017)	Ошибка перечисления значений в ключе реестра

Имя и код ошибки	Описание и причина возникновения ошибки
ERR_PROFILES_NO_PROFILE (0xE0D50018)	При попытке выбора профиля (не в режиме редактирования) в реестре не обнаружено ни одного профиля либо при записи информации о профилях в реестр не было сформировано ни одного профиля
ERR_PROFILES_BAD_CONFIG (0xE0D50019)	Либо в реестре содержится неадекватное (меньше 2) значение параметра "count" обозначающего количество хранилищ для профиля. Либо в конфигурационном файле профиля (cfg.ini) в разделе [ODBC] не задан или задан пустой параметр local.gdbm при параметре local.gdbm_type равном 2
ERR_PROFILES_PROFILE_NOT_FOUND (0xE0D5001A)	Не найден профиль с заданным именем
ERR_PROFILES_PROF_ALREADY_EXISTS (0xE0D5001B)	При попытке добавления нового профиля без флага, разрешающего перезапись, обнаружено, что профиль с таким именем уже есть
ERR_PROFILES_BAD_PROF_INDEX (0xE0D5001C)	Не найден профиль с заданным номером
ERR_PROFILES_FILE_INSTEAD_DIR (0xE0D5001D)	При попытке создания директории (каталога) для хранения профиля обнаружено, что существует файл с таким именем
ERR_PROFILES_AC_DENY_CREATE_DIR (0xE0D5001E)	Недостаточно прав для создания директории (каталога)
ERR_PROFILES_CREATE_DIR_NO_PARENT (0xE0D5001F)	Попытка создать поддиректорию (подкаталог) отсутствующей директории (каталога)
ERR_PROFILES_CREATE_DIR_NO_ROOT (0xE0D50020)	Попытка создать поддиректорию (подкаталог) при отсутствии корня (например, диска)
ERR_PROFILES_DONT_CREATE_DIR (0xE0D50021)	Ошибка создания директории (каталога), не относящаяся к вышеперечисленным
ERR_PROFILES_ODBC (0xE0D50022)	Ошибка вызова функций SQLAllocHandle(), или SQLSetEnvAttr(), или SQLDriverConnect() библиотеки odbc32.dll. Проблемы библиотеки ODBC
ERR_PROFILES_BAD_LDAP_STRING (0xE0D50023)	Ошибка разбора строки LDAP-соединения
ERR_PROFILES_OPEN_MY_STORE (0xE0D50024)	Ошибка функции CertOpenStore() библиотеки Crypt32.dll, открывающей хранилище личных сертификатов
ERR_PROFILES_ENUM_MY_CERTS (0xE0D50025)	Ошибка функции CertEnumCertificatesInStore() библиотеки Crypt32.dll перечисляющей сертификаты из хранилища личных
ERR_PROFILES_GET_CERT_SUBJECT (0xE0D50026)	Ошибка функции CertNameToStr() Crypt32.dll, получающей имя владельца сертификата. Возможно, испорчен сертификат
ERR_PROFILES_NO_MY_CERTS (0xE0D50027)	Не найдено ни одного сертификата в хранилище личных сертификатов
ERR_PROFILES_FILETIME_TO_SYSTIME (0xE0D50028)	Ошибка функции FileTimeToLocalFileTime() или ф-ии FileTimeToSystemTime() библиотеки Kernel32.dll. Возможно, в сертификате указано неадекватное время
ERR_PROFILES_NO_SUBJ_KEY_ID (0xE0D50029)	В сертификате не найдено расширение 'Идентификатор ключа владельца'
ERR_PROFILES_DECODE_OBJECT (0xE0D5002A)	Ошибка функции CryptDecodeObject(), декодирующей объект в ASN1 кодировке. Возможно, испорчен сертификат
ERR_PROFILES_FIND_CERT_BY_KEYID (0xE0D5002B)	Ошибка поиска сертификата ф-ией CertFindCertificateInStore() с параметром CERT_FIND_CERT_ID по идентификатору ключа владельца. Возможно, сертификат отсутствует
ERR_PROFILES_SHOW_CERT (0xE0D5002C)	Ошибка функции CryptUIDlgViewContext(), отображающей сертификат

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД	База данных
ДСЧ	Датчик случайных чисел
КЗИ	Криптографическая защита информации
ЛСП	Локальный справочник пользователя
ОС	Операционная система (Operating System)
ПО	Программное обеспечение
ППИ	Прикладной программный интерфейс
ПСП	Персональный справочник сертификатов
САС	Список аннулированных сертификатов
СКАД	Система криптографической авторизации электронных документов
ССС	Сетевой справочник сертификатов
СУС	Система управления сертификатами
ЦР	Центр регистрации
ЦС	Центр сертификации
ЭП	Электронная подпись (Digital Signature)

ПЕРЕЧЕНЬ ТАБЛИЦ

1	Описание ошибочных ситуаций	74
---	---------------------------------------	----

[illegible][illegible]