

ООО «ВАЛИДАТА»

УТВЕРЖДЕН
ВАМБ.00077-06-ЛУ

«ВАЛИДАТА КЛИЕНТ» ВЕРСИЯ 4

БИБЛИОТЕКА ПРИКЛАДНОГО ПРОГРАММНОГО ИНТЕРФЕЙСА
РАБОТЫ С СЕРТИФИКАТАМИ ДЛЯ C/C++

Руководство программиста

ВАМБ.00077-06 33 01

2020

Аннотация

Данный документ содержит описание библиотеки прикладного программного интерфейса работы с сертификатами для C/C++ для операционной системы (ОС) Microsoft Windows программного комплекса (ПК) ВАНБ.00077-06 «“Валидата Клиент” версия 4» (далее по тексту — ПК «Валидата Клиент»), а также рекомендации по встраиванию и использованию данной библиотеки в прикладном программном обеспечении (ПО).

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

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

Содержание

1	БИБЛИОТЕКА ПРИКЛАДНОГО ПРОГРАММНОГО ИНТЕРФЕЙСА	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
1.3.4	Подготовка среды разработки	11
1.3.5	Параметры конфигурации библиотеки	11
1.4	Описание базовых типов	14
1.5	Описание структур блока памяти	15
1.6	Описание структур идентификации сертификата	15
1.7	Описание структур сертификата	16
1.8	Описание структур САС	21
1.9	Инициализация и деинициализация	22
1.9.1	Структуры инициализации	23
1.9.2	Функции инициализации	25
1.9.3	Функции деинициализации	27
1.10	Получение описаний ошибок	28
1.10.1	Функции получения описаний ошибок	28
1.11	Управление и вызов Мастеров	29
1.11.1	Функции управления и вызова Мастеров	29
1.12	Справочники и профили пользователя	31
1.12.1	Функции справочников и профилей пользователя	31
1.13	Визуализация объектов СУС	34
1.13.1	Функции визуализации объектов СУС	34
1.14	Экспорт и импорт объектов СУС	35
1.14.1	Структуры экспорта и импорта объектов СУС	35
1.14.2	Функции экспорта и импорта объектов СУС	37
1.15	Разбор и получение информации об объектах СУС	38
1.15.1	Структуры разбора и получения информации об объектах СУС	39
1.15.2	Функции разбора и получения информации об объектах СУС	39
1.16	Построение и проверка цепочек объектов СУС	41
1.16.1	Структуры построения и проверки цепочек объектов СУС	43
1.16.2	Функции построения и проверки цепочек объектов СУС	44
1.17	Вычисление хэш-значений	45
1.17.1	Функции вычисления хэш-значений	45
1.18	Вычисление и проверка ЭП хэш-значений	47
1.18.1	Функции вычисления и проверки ЭП хэш-значений	47
1.19	Поиск и перечисление объектов СУС	48
1.19.1	Структуры поиска сертификатов	49
1.19.2	Функции поиска сертификатов	51
1.19.3	Функции перечисления объектов СУС	51
1.20	Вычисление ЭП CMS-сообщений	53
1.20.1	Структуры вычисления ЭП CMS-сообщений	54

1.20.2	Функции блочного вычисления совмещенной ЭП CMS-сообщений	54
1.20.3	Функции потокового вычисления совмещенной ЭП CMS-сообщений	55
1.20.4	Функции блочного вычисления отделенной ЭП CMS-сообщений	56
1.20.5	Функции потокового вычисления отделенной ЭП CMS-сообщений	57
1.21	Проверка ЭП CMS-сообщений	58
1.21.1	Структуры проверки ЭП CMS-сообщений	60
1.21.2	Функции блочной проверки совмещенных ЭП CMS-сообщений	63
1.21.3	Функции потоковой проверки совмещенных ЭП CMS-сообщений	64
1.21.4	Функции блочной проверки отделенных ЭП CMS-сообщений .	65
1.21.5	Функции потоковой проверки отделенных ЭП CMS-сообщений	67
1.22	Зашифрование CMS-сообщений	68
1.22.1	Структуры зашифрования CMS-сообщений	69
1.22.2	Функции блочного зашифрования CMS-сообщений	71
1.22.3	Функции потокового зашифрования CMS-сообщений	71
1.23	Расшифрование CMS-сообщений	72
1.23.1	Структуры расшифрования CMS-сообщений	73
1.23.2	Функции блочного расшифрования CMS-сообщений	73
1.23.3	Функции потокового расшифрования CMS-сообщений	74
1.24	Преобразование совмещенных и отделенных ЭП	76
1.24.1	Функции блочного преобразования отделенных ЭП в совмещенные	76
1.24.2	Функции блочного преобразования совмещенных ЭП в отделенные	76
1.24.3	Функции потокового преобразования совмещенных ЭП в отделенные	77
1.25	Получение информации о CMS-сообщениях	78
1.25.1	Структуры получения информации о CMS-сообщениях	78
1.25.2	Функции блочного получения информации о CMS-сообщениях	82
1.25.3	Функции потокового получения информации о CMS-сообщениях	83
1.26	Простановка и проверка штампов времени	84
1.26.1	Структуры простановки и проверки штампов времени	85
1.26.2	Функции блочной простановки и проверки штампов времени .	86
1.26.3	Функции потоковой простановки и проверки штампов времени	89
1.27	Получение online-статуса сертификата	92
1.27.1	Структуры получения online-статуса сертификата	93
1.27.2	Функции получения online-статуса сертификата	94
1.28	Протокол безопасности транспортного уровня TLS 1.2	96
1.28.1	Функции протокола безопасности транспортного уровня TLS 1.2	97
1.29	Преобразование в формат и из формата Base64	100
1.29.1	Функции преобразования в формат и из формата Base64	100
1.30	Выработка случайного числа заданной длины	101
1.30.1	Функции выработки случайного числа заданной длины	101
1.31	Выделение и освобождение памяти	101
1.31.1	Функции выделения и освобождения памяти	102
1.32	Описание конфигурационного файла pk1.conf	104

1.33 Описание XML-шаблона и XML-запроса	105
2 ОПИСАНИЕ ОШИБОЧНЫХ СИТУАЦИЙ	109
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ	118
ПЕРЕЧЕНЬ РИСУНКОВ	119
ПЕРЕЧЕНЬ ТАБЛИЦ	120

1 БИБЛИОТЕКА ПРИКЛАДНОГО ПРОГРАММНОГО ИНТЕРФЕЙСА

1.1 Назначение библиотеки

Библиотека прикладного программного интерфейса работы с сертификатами для C/C++ для операционной системы (ОС) Microsoft Windows (далее по тексту - библиотека) является составной частью программного комплекса (ПК) ВАМБ.00077-06 «Валидата Клиент» версия 4» (далее по тексту - ПК «Валидата Клиент») и обеспечивает функции 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.

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», которые в свою очередь соответствуют одноименной спецификации Технического комитета № 26.

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 «Форматы сообщений, защищенных криптографическими методами», которые в свою очередь соответствуют спецификации «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)» Технического комитета № 26.

Реализация протокола безопасности транспортного уровня TLS 1.2 библиотеки соответствуют Рекомендациям по стандартизации Р 1323565.1.020-2018 «Использование российских криптографических алгоритмов в протоколе безопасности транспортного уровня (TLS 1.2)», которые в свою очередь соответствует одноименной спецификации Технического комитета № 26.

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 предназначена для встраивания ПК «Валидата Клиент» в прикладные системы. Требования к аппаратно-программной среде, в которой функционирует библиотека, приведены в документе ВАМБ.00077-06 31 01 «"Валидата Клиент" версия 4. Описание применения».

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 (поток).

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.

Примечания

1 Шифрование файла или области памяти как по ГОСТ 28147-89, так и по ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015 (блочные шифры «Магма» и «Кузнечик»), выполняется так называемым анонимным способом. При таком способе шифрования для вычисления ключей согласования Диффи-Хеллмана не используются закрытый и открытый ключи шифрования отправителя, а используется временная (эфемерная) ключевая пара. При анонимном способе шифрования, в том числе при использовании контроля целостности с помощью имитовставки, не обеспечиваются ни защита от навязывания данных, ни подтверждение их авторства - для этих целей необходимо использовать ЭП.

2 При шифровании файла или области памяти по ГОСТ Р 34.12-2015, ГОСТ Р 34.13-2015 (блочный шифр «Магма») в режиме гаммирования максимальный объем шифруемых данных ограничен 32 Гбайтами.

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 учетом 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 **zпки1.dll** библиотеки всегда имеет такую же битность, как и ПО ПК «Валидата Клиент», в состав которого он входит. Битность исполняемого модуля **zпки1.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, CAC 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, CAC 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, CAC, запросами PKCS#10, запросами на аннулирование, содержащимися в хранилищах ПСП, ЛСП и (опционально) СССР. Единственным исключением из этого правила является возможность загрузки сертификатов ЦС второго и более низких уровней из точек AIA и CAC 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:

- 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;

- библиотека предназначена для использования в приложениях либо написанных на языках программирования C/C++, либо совместимых с ними по форматам данных и параметрам вызовов функций. При использовании библиотеки в приложениях, написанных на других языках программирования, должно выполняться требуемое преобразование форматов данных и параметров вызовов функций.

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:

- **zпки1.dll** – модуль динамической библиотеки;
- **zпки1.lib** – модуль библиотеки для линковки с динамической библиотекой;
- **vcert1.h** – файл заголовков с определениями констант, структур и прототипов функций библиотеки;
- **vcerrerr.h** – файл заголовков с определениями кодов возврата (кодов ошибок) функций библиотеки;
- **misc.c, tool.c, util.c, util.h** - файлы с исходными текстами утилиты командной строки.

1.3.4 Подготовка среды разработки

При компиляции приложений необходимо использовать макро **__** для того, чтобы размеры структур переменной длины - в частности, **altname_t** - вычислялись корректно.

В ОС Microsoft Windows все функции библиотеки объявлены как **WINAPI**, таким образом для всех функций библиотеки используется соглашение о вызовах функций языка "Pascal".

В качестве идентификатора родительского окна в ОС Microsoft Windows должно использоваться значение, возвращаемое системной функцией **GetActiveWindow()**.

В процессе разработки прикладного ПО может возникнуть необходимость детального анализа ошибочной ситуации, возникшей при вызове функций библиотеки. Для решения этой задачи библиотека поддерживает возможность включения отладочного протоколирования. Для включения отладочного протоколирования следует создать переменную окружения (Environment Variable) **pki1dbgvlvl**, установить ее значение в **0x80** и перезапустить процесс прикладного ПО.

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

1.3.5 Параметры конфигурации библиотеки

Библиотека позволяет выполнять настройки (изменять значения параметров конфигурации), предназначенные для адаптации к потребностям прикладного

ПО, а также для увеличения производительности при использовании большого количества (1000000 и более) сертификатов.

Значения параметров конфигурации библиотеки хранятся в двух ветках Реестра - пользователя **HKCU\SOFTWARE\Validata\zpci\Parameters** и компьютера **HKLM\SOFTWARE\Validata\zpci\Parameters**. Чтение значений параметров производится вначале всегда из ветки пользователя, и далее (только в том случае, если значение не было считано) - из ветки компьютера.

Ниже (Таблица 1) приведено описание параметров конфигурации библиотеки. Чтение конфигурационных параметров выполняется при первом получении контекста библиотеки после запуска прикладного ПО, поэтому для вступления в силу изменений, внесенных в значения параметров конфигурации, прикладное ПО необходимо перезапустить.

Таблица 1 - Описание параметров конфигурации

Параметр	Описание
PkiLdapTimeout	Таймаут (в секундах), используемый при выполнении операций подключения и поиска объектов СУС в CCC по протоколу LDAP (тип: REG_DWORD). Значение параметра должно быть ≥ 0 (по умолчанию равно 10), при задании 0 таймаут операции подключения составляет 120 секунд, а таймаут операции поиска не используется
PkiLdapPageSize	Размер страницы (в записях), используемый при выполнении операции поиска объектов СУС в CCC по протоколу LDAP (тип: REG_DWORD). Значение параметра должно быть ≥ 0 (по умолчанию равно 50), при задании 0 постраничный поиск в CCC не используется
PkiHttpTimeout	Таймаут (в секундах), используемый при выполнении операций подключения и поиска объектов в точках AIA и CDP по протоколу HTTP (тип: REG_DWORD). Значение параметра должно быть > 0 (по умолчанию равно 60)
PkiODBCUseMars	Включение многопоточной обработки (Multiple Active Results Sets, MARS) для ЛСП, расположенных в БД ODBC (тип: REG_DWORD). Значение параметра должно быть 0 или 1 (по умолчанию равно 0, многопоточная обработка выключена). При включении многопоточной обработки для КАЖДОГО задействованного в работе библиотеки DSN должны выполняться следующие условия: 1 - либо БД данного DSN находится под управлением СУБД Microsoft SQL Server и в данном DSN используется ODBC драйвер ODBC Driver for SQL Server или SQL Server Native Client , либо БД данного DSN находится под управлением СУБД Oracle или PostgreSQL; 2 - либо БД данного DSN находится под управлением СУБД Microsoft SQL Server и значение параметра Mars_Connection данного DSN (типа REG_SZ) установлено в Yes , либо БД данного DSN находится под управлением СУБД Oracle или PostgreSQL
PkiODBCUsername PkiODBCPassword	Имя и пароль пользователя для аутентификации с СУБД (тип: REG_SZ). В случае, когда для доступа к ЛСП, находящегося под управлением СУБД, требуется проведение аутентификации, данные параметры позволяют задать имя и пароль аутентифицируемого пользователя. Задание значений имени и пароля пользователя является обязательным, если ЛСП находится под управлением СУБД Oracle или PostgreSQL
PkiODBCConnectStr	Строка соединения (без идентификатора DSN), используемая при подключении к СУБД (тип: REG_SZ). В случае, если данный параметр задан, значения параметров PkiODBCUsername и PkiODBCPassword игнорируются, а имя и пароль пользователя берутся из заданной строки соединения
PkiUriAIAExternal<N> (строковое значение внешнего URI) PkiUriAIAPInternal<N> (строковое значение внутреннего URI) ($0 \leq N < M$, M - общее количество заменяемых URI точек AIA)	Настраиваемые параметры, позволяющие включить механизм замены URI точек AIA (тип: REG_SZ). Значения параметров могут быть любыми, с учетом использования символов из набора ASN.1 IA5 (ASCII). Во время построения цепочки объекта СУС, при загрузке сертификата ЦС из точки AIA, URI данной точки ищется в списке внешних URI точек AIA. При нахождении этого URI в списке его значение заменяется на значение внутреннего URI, заданного для данной точки AIA, которое, в свою очередь, используется для загрузки сертификата ЦС
PkiUriCDPExternal<N> (строковое значение внешнего URI) PkiUriCDPInternal<N> (строковое значение внутреннего URI) ($0 \leq N < M$, M - общее количество заменяемых URI точек CDP)	Настраиваемые параметры, позволяющие включить механизм замены URI точек CDP (тип: REG_SZ). Значения параметров могут быть любыми, с учетом использования символов из набора ASN.1 IA5 (ASCII). Во время построения цепочки объекта СУС, при загрузке CAC из точки CDP, URI данной точки ищется в списке внешних URI точек CDP. При нахождении этого URI в списке его значение заменяется на значение внутреннего URI, заданного для данной точки CDP, которое, в свою очередь, используется для загрузки CAC
PkiCacheLowMark PkiCacheHighMark	Нижняя и верхняя ватерлинии (в объектах) кэша соответственно (тип: REG_DWORD). При размере кэша, превышающем верхнюю ватерлинию, производится удаление из кэша наименее используемых объектов СУС до достижения равенства размера кэша и нижней ватерлинии. Значения обоих параметров должны удовлетворять условию $0 < PkiCacheLowMark < PkiCacheHighMark$ (по умолчанию равны 0, ватерлинии не используются), при других значениях ватерлинии не используются

Параметр	Описание
Pk1lCacheSbKIdIndex	Включение индексирования идентификаторов ключей владельцев объектов СУС, находящихся в кэше (тип: REG_DWORD). При включенном индексировании поиск по идентификатору ключа владельца будет выполняться исключительно по индексу. Значение параметра должно быть 0 или 1 (по умолчанию равно 0, индексирование выключено). Включение индексирования обеспечивает повышение производительности операций поиска по идентификатору ключа владельца, но ухудшает производительность операций добавления в кэш и удаления из кэша
Pk1lCacheKeyIdIndex	Включение индексирования идентификаторов закрытых ключей объектов СУС, находящихся в кэше (тип: REG_DWORD). При включенном индексировании поиск по идентификатору закрытого ключа будет выполняться исключительно по индексу. Значение параметра должно быть 0 или 1 (по умолчанию равно 0, индексирование выключено). Включение индексирования обеспечивает повышение производительности операций поиска по идентификатору закрытого ключа, но ухудшает производительность операций добавления в кэш и удаления из кэша
Pk1lStackHashes	Количество корзин для хранения объектов СУС, находящихся в кэше (тип: REG_DWORD). Кэш представляет собой упорядоченный список корзин, каждая из которых в свою очередь представляет собой упорядоченный список объектов СУС. Доступ к корзинам синхронизируется между потоками прикладного процесса, индивидуально для каждой корзины. Для повышения производительности рекомендуется наличие в корзине, в среднем, не более 1024 объектов СУС. Значение параметра должно удовлетворять условию $1 \leq \text{Pk1lStackHashes} \leq 4095$ (по умолчанию равно 127). Увеличение количества корзин приводит к повышению производительности и параллелизма выполнения, однако при этом увеличиваются требования к используемым ресурсам
Pk1lGroup<G>AffinityL Pk1lGroup<G>AffinityH (G - номер процессорной группы, $0 \leq G \leq 7$)	Нижняя и верхняя маски привязки к логическим процессорам процессорной группы G (тип: REG_DWORD). Значения параметров могут быть любыми (по умолчанию равны 0, привязки к логическим процессорам отсутствуют). Установленный в масках бит с номером N, $0 \leq N \leq 63$, означает, что потоки могут использовать логический процессор с номером N данной процессорной группы. Например, для двух процессорных групп по 40 процессоров в каждой задание значений масок Pk1lGroup0AffinityL = 0xffffffff, Pk1lGroup0AffinityH = 0x000000ff, Pk1lGroup1AffinityL = 0xffffffff, Pk1lGroup1AffinityH = 0x000000ff означает, что могут использоваться все 80 логических процессоров обеих групп

1.4 Описание базовых типов

`typedef uint32_t error_status_t`

Статус, код возврата, код ошибки.

`typedef void * context_t`

Контекст сессии работы с библиотекой, возвращается функцией инициализации.

`typedef void * strhash_t`

Контекст выполнения потоковых функций вычисления хэш-значения данных.

`typedef void * enuobj_t`

Контекст перечисления объектов справочников (персонального, локального, сетевого).

`typedef void * strcms_t`

Контекст выполнения потоковых функций CMS-сообщений.

`typedef void * tls_t`

Контекст защищенного канала (сеанса связи) по протоколу TLS.

`typedef unsigned char * string_t`

Строка, заканчивающаяся символом '\0' - по умолчанию, используется кодировка Windows-1251.

`typedef uint32_t certinfo_t`

Требуемая или возвращенная информация о сертификате - поля сертификата, битовая маска из констант `FIELD_XXX`.

`typedef uint32_t crlinfo_t`

Требуемая или возвращенная информация о САС - поля САС, битовая маска из констант `FIELD_CRL_XXX`.

`typedef uint32_t keyusage_t`

Возможные области использования открытого ключа - битовая маска из констант `KEYUSAGE_XXX`.

`typedef uint32_t date_t`

Время в секундах, прошедшее с 00:00 01.01.1970 UTC - данный тип аналогичен 32-битному `time_t`.

`typedef uint32_t flag_t`

Параметры, модификаторы или флаги выполнения функций.

1.5 Описание структур блока памяти

Структура `mem_blk_t`

Непрерывный блок памяти (блок данных):

– `uint32_t len`

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

– `unsigned char * buf`

Указатель на блок памяти, может быть равен `NULL` только если длина блока равна 0.

Структура блока памяти используется для обмена бинарными данными между приложением и библиотекой. При передаче данных из приложения в библиотеку структура должна содержать корректный указатель на блок памяти и его длину. В случае отсутствия передаваемых данных или при получении данных из библиотеки указатель и длина должны быть обнулены.

1.6 Описание структур идентификации сертификата

Структура `issuer_and_serial_t`

Пара издателя и серийного номера сертификата:

– `string_t issuer`

Строка с X.500-именем издателя сертификата.

– `string_t serialNumber`

Строка с серийным номером сертификата.

*Примечание - В настоящее время структура, предназначенная для идентификации сертификата (уникальный идентификатор сертификата), используется исключительно в целях определения контекста выполнения функции библиотеки - для определения сессии КС, в контексте которой необходимо выполнить требуемую функцию. При использовании локальной библиотеки указатель на объект структуры **certid_t**, обычно поименованный как **mycert**, необходимо обнулить - это связано с тем, что по контексту локальной библиотеки однозначно определяется сертификат с закрытым ключом (рабочий сертификат).*

Структура certid_t

Уникальный идентификатор сертификата:

– **idchoice_t type**

Выбор способа идентификации: ID_ISSUER_AND_SERIAL - по паре издателя и серийного номера сертификата; ID_KEYID - по идентификатору закрытого ключа, соответствующего данному сертификату; ID_CERTHASH - по хэш-значению пары имени издателя и серийного номера сертификата.

– **issuer_and_serial_t id.ias**

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

– **string_t id.keyId**

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

– **mem_blk_t id.certHash**

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

1.7 Описание структур сертификата

Каждый сертификат может включать в себя список регламентов (политик) использования сертификата, которые содержатся в дополнении (расширении) **certificatePolicies**. Каждый регламент использования сертификата представляют собой идентификатор, который предназначен определять:

- область использования сертификата и соответствующего ему закрытого ключа;
- комплекс организационно-технических мероприятий, связанных с жизненным циклом объектов СУС (распространение ПО, регистрация, сертификация, компрометация и т. п.).

Каждый регламент использования сертификата определяет систему, в которой разрешено использовать данный сертификат и соответствующий ему закрытый ключ. Одновременное наличие в сертификате нескольких регламентов позволяет использовать его в нескольких системах, соответствующих данным регламентам.

С целью определения политики использования сертификата (разделения прав и полномочий субъекта/объекта), сертификат может включать в себя список расширенных использований открытого ключа, которые содержатся

в дополнении **extendedKeyUsage**. Каждое расширенное использование открытого ключа представляет собой идентификатор, который определяет один или несколько типов операций, которые можно выполнять с использованием данного сертификата и соответствующего ему закрытого ключа.

Структура altname_t

Альтернативное имя издателя или владельца сертификата:

– string_t **emailAddress**

Строка с адресом электронной почты RFC 822.

– string_t **DNS**

Строка с именем системы имен доменов DNS.

– string_t **URI**

Строка с уникальным адресом ресурса URI.

– string_t **IP**

Строка с адресом IP протокола.

– string_t **organizationName**

Строка с наименованием организации.

– string_t **registredAddress**

Строка с зарегистрированным адресом.

– string_t **surname**

Строка с Ф.И.О. владельца сертификата.

– string_t **businessCategory**

Строка с должностью владельца сертификата.

– string_t **telephoneNumber**

Строка с номером телефона владельца сертификата.

– string_t **description**

Строка с описанием в свободной форме.

– string_t **account_number**

Строка с номером расчетного счета.

– string_t **bank_id**

Строка с банковским идентификационным кодом (БИК).

– string_t **physicalDelivery**

Строка с почтовым адресом.

– string_t **exchange_address**

Строка с адресом Microsoft Exchange.

– string_t **notes_address**

Строка с адресом Lotus Notes.

Структура policy_t

Регламент (политика) использования сертификата:

– string_t **oid**

Строка объектного идентификатора (OID), идентифицирующего регламент сертификата.

Структура extkeyusage_t

Расширенная область использования открытого ключа сертификата:

– string_t **oid**

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

Структура extension_t

Дополнение (расширение) X.509 сертификата:

– string_t **oid**

Строка объектного идентификатора (OID), идентифицирующего дополнение сертификата.

– uint32_t **type**

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

– uint32_t **critical**

Признак критичности дополнения.

– uint32_t **len**

Длина блока памяти дополнения в байтах.

– unsigned char * **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.

Например, для дополнения "Идентификатор ключа владельца" с объектным идентификатором (OID) 2.5.29.14 в блоке памяти будет содержаться хэш-значение открытого ключа сертификата, вычисленное по алгоритму FIPS 180-1 SHA-1.

Структура certificate_t

Структура, описывающая сертификат:

– certinfo_t **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_t **issuer**

Поле с X.500-именем издателя сертификата в виде строки (при задании FIELD_ISSUER в поле fields), например CN=TestUser,DC=X509,DC=RU. Если дополнительно в поле fields задан FIELD_X500_NAME_OID_FORMAT, то вместо названий частей X.500-имени издателя сертификата возвращаются соответствующие им объектные идентификаторы (OID), например 2.5.4.3=TestUser,0.9.2342.19200300.100.1.25=X509,0.9.2342.19200300.100.1.25=RU.

– string_t **serialNumber**

Поле с серийным номером сертификата в виде стро-

ки (при задании `FIELD_SERIAL` в поле `fields`), например 00:01:02:03:04:05:06:07:08:09:0A:0B:0C:0D:0E:0F.

– `string_t subject`

Поле с X.500-именем владельца сертификата в виде строки (при задании `FIELD_SUBJECT` в поле `fields`). Если дополнительно в поле `fields` задан `FIELD_X500_NAME_OID_FORMAT`, то вместо названий частей X.500-имени владельца сертификата возвращаются соответствующие им объектные идентификаторы (OID).

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_t algorithm`

Поле со строкой алгоритма открытого ключа сертификата (при задании `FIELD_ALGORITHM` в поле `fields`): ГОСТ Р 34.10-2001 - "ГОСТ R 34.10-2001"; ГОСТ Р 34.10-2012 (256 бит) - "ГОСТ R 34.10-2012 (256 bit)"; ГОСТ Р 34.10-2012 (512 бит) - "ГОСТ R 34.10-2012 (512 bit)". Если, дополнительно, в поле `fields` задан `FIELD_ALGORITHM_OID_FORMAT`, то возвращается объектный идентификатор (OID) алгоритма открытого ключа сертификата в текстовом виде: ГОСТ Р 34.10-2001 - "1.2.643.2.2.19" или "1.3.6.1.4.1.3670.1.9"; ГОСТ Р 34.10-2012 (256 бит) - "1.2.643.7.1.1.1.1"; ГОСТ Р 34.10-2012 (512 бит) - "1.2.643.7.1.1.1.2".

– `date_t notBefore`

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

– `date_t notAfter`

Поле с датой окончания действия сертификата (при задании `FIELD_`

NOTAFTER в поле fields).

– `keyusage_t` **keyUsage**

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

- **KEYUSAGE_DIGITAL_SIGNATURE** - использование открытого ключа для проверки ЭП документов, отличных от сертификатов и САС;
- **KEYUSAGE_NON_REPUDIATION** - означает невозможность отказа подписавшего лица от своего действия;
- **KEYUSAGE_KEY_ENCIIPHERMENT** - использование для шифрования закрытых или секретных ключей;
- **KEYUSAGE_DATA_ENCIIPHERMENT** - использование для шифрования пользовательских данных без методов симметричной криптографии;
- **KEYUSAGE_KEY_AGREEMENT** - использование в целях вычисления ключей согласования;
- **KEYUSAGE_KEY_CERT_SIGN** - использование открытого ключа для проверки ЭП сертификатов;
- **KEYUSAGE_CRL_SIGN** - использование открытого ключа для проверки ЭП САС;
- **KEYUSAGE_ENCIIPHER_ONLY** - использование в целях вычисления ключей согласования только для зашифрования данных;
- **KEYUSAGE_DECIPHER_ONLY** - использование в целях вычисления ключей согласования только для расшифрования данных.

– `date_t` **notBeforePrivate**

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

– `date_t` **notAfterPrivate**

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_t` **keyId**

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

– `uint32_t` **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).

– uint32_t **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).

– uint32_t **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).

При заполнении шаблона сертификата для поиска или зашифрования по идентификатору ключа владельца необходимо включить в данный массив дополнение "Идентификатор ключа владельца" с объектным идентификатором (OID) 2.5.29.14 и ASN.1 типом дополнения 4 (OctetString). В блоке памяти включенного дополнения должно содержаться хэш-значение открытого ключа сертификата, вычисленное по алгоритму FIPS 180-1 SHA-1.

– 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). Значение данного поля уникально для каждого сертификата и является первичным ключом при поиске сертификата в справочниках.

1.8 Описание структур САС

Структура revcert_t

Аннулированный сертификат:

– string_t **serialNumber**

Поле с серийным номером аннулированного сертификата в виде строки.

– date_t **revtime**

Поле со временем аннулирования сертификата.

– int32_t **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:

– `crlinfo_t` **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_t` **issuer**

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

– `date_t` **lastUpdate**

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

– `date_t` **nextUpdate**

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

– `uint32_t` **number**

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

– `uint32_t` **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 Инициализация и деинициализация

В ходе выполнения упрощенной или расширенной инициализации контекста локальной библиотеки соответствующая функция:

- производит чтение конфигурационных параметров библиотеки и выполняет ее инициализацию;

- подключается к хранилищам ПСП, ЛСП и (опционально) ССС, заданным в настройках профиля пользователя;
- в хранилищах производит поиск сертификата пользователя (рабочего сертификата), на котором защищен ПСП;
- если не указано противное, производит обновление всех САС, найденных в ЛСП, из точек CDP;
- проверяет ЭП и использование ПСП, строит и проверяет цепочку сертификации рабочего сертификата;
- в случае загрузки закрытого ключа проверяет на соответствие ему рабочий сертификат.

1.9.1 Структуры инициализации

Структура `local_param_t`

Параметры инициализации контекста локальной библиотеки:

- `size_t size`

Поле с размером структуры в байтах, должно быть установлено в `sizeof(local_param_t)`.

- `flag_t flag`

Поле с маской (побитовым ИЛИ) флагов инициализации контекста библиотеки:

- **FLAG_INIT_DISABLECRLUPDATE** - не выполнять автоматическое обновление САС, найденных в ЛСП, при инициализации контекста. Если данный флаг не установлен, при инициализации контекста перебираются все САС из ЛСП, и для каждого из них загружаются обновления из указанных в данном САС точек CDP;

- **FLAG_INIT_EXPIRINGOBJECTUI** - показывать объекты с истекающим сроком действия при инициализации контекста. При задании данного флага производится показ сертификата с закрытым ключом (рабочего сертификата) с истекающим сроком действия закрытого ключа, сертификатов ЦС с истекающими сроками действия закрытых ключей и самих сертификатов, а также истекающих САС. Интервалы истечения объектов задаются в ключе Реестра **HKCU\SOFTWARE\Validata\zcs**. Значение **TimeWarnExpiredCert** (типа `REG_DWORD`) указывает интервал истечения закрытого ключа рабочего сертификата и закрытых ключей и сертификатов ЦС в днях (по умолчанию - 30 дней). Значение **TimeWarnExpiredCrl** (типа `REG_DWORD`) указывает интервал истечения САС в днях (по умолчанию - 14 дней);

- **FLAG_INIT_DISABLELDAPUSAGE** - не подключаться к ССС при инициализации. При установке данного флага поиск сертификатов или перечисление сертификатов и САС в ССС будут отключены;

- **FLAG_INIT_DISABLECACHESAVE** - не сохранять проверенные сертификаты и САС, находящиеся в кэше библиотеки, в ЛСП при деинициализации контекста. Если данный флаг не установлен, кэшированные сертификаты и САС будут сохранены в ЛСП при деинициализации контекста -

такой подход повышает производительность за счет минимизации количества операций записи;

- **FLAG_INIT_CERTSTOREPROFILE** - использовать настройки профилей пользователя ПК "Справочник сертификатов" из Реестра вместо конфигурационного файла **pki1.conf**;

- **FLAG_INIT_FORCECERTCACHING** - принудительно кэшировать сертификаты, предназначенные для шифрования, при инициализации контекста. Данный флаг поддерживается только для ЛСП, расположенных в БД ODBC, при включенной многопоточной обработке (см. описание параметра **PkiODBCUseMars** в таблице 1);

- **FLAG_INIT_ALLOWAIACDPUSAGE** - разрешить доступ к точкам AIA и CDP для загрузки сертификатов ЦС и САС при построении цепочек. Доступ к точкам будет осуществляться в следующих случаях:

- сертификат ЦС или САС, необходимый для построения цепочки, не найден в ПСП и ЛСП;

- срок действия САС, необходимого для построения цепочки, уже истек;

- **FLAG_INIT_USESILENTKEYLOAD** - не выдавать пользовательский интерфейс при загрузке закрытых ключей. Если для загрузки закрытого ключа вмешательство пользователя необходимо - для ввода пароля или ПИН-кода, для установки ключевого носителя и т.п. - операция загрузки завершится с ошибкой;

- **FLAG_INIT_USEVERIFYCONTEXT** - инициализировать контекст без доступа к закрытому ключу (так называемый контекст проверки). Если данный флаг установлен, то любая попытка использовать закрытый ключ с полученным контекстом библиотеки - для вычисления ЭП, для расшифрования и т.п. - завершится с ошибкой.

– const char * **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.

– const char * **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).

– *const char** **ldap**

Поле с путем (URI) к ССС следующего вида:

- **ldap://[simple/|negotiate/][user[:pass]@]host[:389]/DC=RU** - ССС, 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 ССС. Модификатор **simple/** указывает, что следует использовать простой механизм аутентификации с ССС, 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 ССС.

– *const char** **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.

К каждому контексту библиотеки, полученному посредством вызова упрощенной или расширенной функции инициализации, привязана структура (кэш), хранящая проверенные сертификаты и САС. Кэш является динамической структурой, расположенной в виртуальной памяти процесса, при этом у любых двух различных контекстов библиотеки эти структуры различны.

Объект (сертификат или САС) попадает в кэш только в случае полной и успешной проверки цепочки сертификации данного объекта. Например, если при вызове функции зашифрования выполняется поиск сертификатов получателей зашифрованного сообщения в ССС, то те найденные сертификаты получателей, чьи цепочки были успешно проверены - а также сертификаты ЦС и САС, участвовавшие в проверке данных цепочек - попадут в кэш. Кэш предназначен для повышения производительности при поиске и проверке объектов, поскольку, во-первых, объекты ищутся по первичному ключу в кэше быстро, и во-вторых, поскольку цепочка объекта, найденного в нем, не будет проверяться повторно.

error_status_t VCERT1API **VCERT_InitMinimal** (context_t * phCtx)

Функция инициализации минимального контекста библиотеки

Аргументы:

- **phCtx** (out) указатель на инициализируемый контекст.

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.

error_status_t VCERT1API **VCERT_Initialize** (string_t profile, flag_t flag, context_t * phCtx)

Упрощенная функция инициализации контекста библиотеки

Аргументы:

– **profile** (in) указатель на строку с именем профиля из настроек конфигурационного файла **pki1.conf** (см. описание в подразделе 1.32), 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 данный указатель может быть равен 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 см. описание в п.1.9.1, иначе значение зарезервировано и не используется);

– **phCtx** (out) указатель на инициализируемый контекст.

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_InitializeEx** (crypt_prov_t prov, void * param, context_t * phCtx)

Расширенная функция инициализации контекста библиотеки

Аргументы:

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

– **param** (in) указатель на структуру параметров инициализации контекста (структуру local_param_t для локальной библиотеки);

– **phCtx** (out) указатель на инициализируемый контекст.

Возвращаемые значения:

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

1.9.3 Функции деинициализации

error_status_t VCERT1API **VCERT_Uninitialize** (context_t hCtx)

Функция деинициализации (освобождения) контекста библиотеки

Аргументы:

– **hCtx** (in) освобождаемый контекст.

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

`char * VCERT1API VCERT_GetErrorText (error_status_t rslt, char * merr, size_t mlen)`

Функция получения текстового описания ошибки по ее коду

Аргументы:

- ***rslt*** (in) код ошибки;
- ***merr*** (out) указатель на буфер для записи текстового описания ошибки (строка описания завершается символом `'\0'`);
- ***mlen*** (in) размер буфера для записи текстового описания ошибки (размер должен быть не менее 128 байт).

Возвращаемые значения:

- **указатель на буфер, равный *merr*** с текстовым описанием ошибки, ***NULL*** - в случае ошибки.

`error_status_t VCERT1API VCERT_GetErrorLineEx (context_t hCtx, mem_blk_t * error, mem_blk_t * file, uint32_t * line, mem_blk_t * data)`

Функция получения стека внутренних ошибок библиотеки

Аргументы:

- ***hCtx*** (in) контекст библиотеки (при анализе ошибки, возникшей в функции инициализации, разрешено подавать значение контекста равным `NULL`);
- ***error*** (out) блок памяти с кодом внутренней ошибки и ее текстовым описанием (освобождается функцией **`VCERT_FreeMem()`** - см. описание в п. 1.31.1);
- ***file*** (out) блок памяти с именем файла исходных текстов, в котором возникла внутренняя ошибка (освобождается функцией **`VCERT_FreeMem()`** - см. описание в п. 1.31.1);
- ***line*** (out) номер строки файла исходных текстов, в которой возникла внутренняя ошибка;
- ***data*** (out) блок памяти с расширенными данными внутренней ошибки (освобождается функцией **`VCERT_FreeMem()`** - см. описание в п. 1.31.1).

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***.

1.11 Управление и вызов Мастеров

1.11.1 Функции управления и вызова Мастеров

`error_status_t VCERT1API VCERT_ControlEx (context_t hCtx, void * hWnd, vcert_cmd_t cmd, mem_blk_t * idat, mem_blk_t * odat, flag_t flag)`

Функция управления контекстом и библиотекой, вызова Мастеров

Аргументы:

– **hCtx** (in) контекст библиотеки;

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

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

• **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**. Для данной команды использование минимального контекста не допускается, а параметры **idat**, **flag** не используются;

• **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 1.33). 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) (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);

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

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_UpdateCRLs** (context_t hCtx)

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**.

error_status_t VCERT1API **VCERT_CreateXmlRequest** (context_t hCtx, flag_t flag, certificate_t * reqtmpl, mem_blk_t * xmlreq)

Функция создания парных ключей и XML запроса по шаблону сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **flag** (in) маска (побитовое ИЛИ) флагов формирования XML-запроса:
 - **FLAG_XML_REQUEST_GOST_R_34_10_12_256** - выполнить генерацию закрытого ключа для сертификата в квалифицированном формате по ГОСТ Р 34.10-2012 (256 бит);
 - **FLAG_XML_REQUEST_GOST_R_34_10_12_512** - выполнить генерацию закрытого ключа для сертификата в квалифицированном формате по ГОСТ Р 34.10-2012 (512 бит);
 - **FLAG_XML_REQUEST_CARRIER_GENERATION** - при генерации неизвлекаемого закрытого ключа на ФКН *vdToken* формировать этот ключ внутри ФКН с использованием внутреннего ДСЧ последнего;
- **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 (см. описание в подразделе 1.33) (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

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

`error_status_t VCERT1API VCERT_CreateCertificateStore (context_t hCtx, mem_blk_t * certs, uint32_t ncert, mem_blk_t * crls, uint32_t ncrl, const char * pstore, const char * localstore, flag_t flag)`

Функция формирования ПСП и ЛСП из отдельных сертификатов и САС

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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** \geq 0);

– **psestore** (out) путь (URI) к создаваемому ПСП (см. описание в п. 1.9.1). 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 (см. описание в п. 1.9.1). 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.

error_status_t VCERT1API **VCERT_CreateRegistryProfile** (context_t hCtx, string_t profile, const char * storepath, flag_t flag)

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

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **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.

error_status_t VCERT1API **VCERT_CreateRegistryProfileEx** (context_t hCtx, string_t profile, const char * psestore, const char * localstore, const char * ldapstore, flag_t flag)

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

Аргументы:

– **hCtx** (in) контекст библиотеки;

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

- **psestore** (in) путь (URI) к ПСП профиля пользователя (см. описание в п. 1.9.1);
- **localstore** (in) путь (URI) к ЛСП профиля пользователя (см. описание в п. 1.9.1);
- **ldapstore** (in) путь (URI) (опциональный) к ССС профиля пользователя (см. описание в п. 1.9.1);
- **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.

error_status_t VCERT1API **VCERT_QueryRegistryProfile** (context_t hCtx, string_t profile, uint32_t length, uint32_t index)

Функция получения информации о профиле по индексу

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **profile** (out) буфер для имени профиля - строки, заканчивающейся символом '\0';
- **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.

error_status_t VCERT1API **VCERT_QueryRegistryProfileEx** (context_t hCtx, mem_blk_t * profile, mem_blk_t * psestore, mem_blk_t * localstore, mem_blk_t * ldapstore, uint32_t index)

Расширенная функция получения информации о профиле по индексу

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **profile** (out) блок памяти для записи имени профиля - строки, заканчивающейся символом '\0' (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **psestore** (out) блок памяти для записи пути (URI) к ПСП - строки, заканчивающейся символом '\0' (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **localstore** (out) блок памяти для записи пути (URI) к ЛСП - строки, заканчивающейся символом '\0' (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **ldapstore** (out) блок памяти для записи пути (URI) к ССС - строки, закан-

чивающейся символом '\0' (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);

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

Возвращаемые значения:

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

1.13 Визуализация объектов СУС

1.13.1 Функции визуализации объектов СУС

error_status_t VCERT1API **VCERT_ShowCertificate** (context_t hCtx mem_blk_t * cert)

Функция визуализации сертификата

Аргументы:

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

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_ShowCertificateEx** (context_t hCtx, void * hWnd, mem_blk_t * cert)

Расширенная функция визуализации сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **hWnd** (in) идентификатор родительского окна (см. описание в п. 1.11.1);
- **cert** (in) блок памяти с сертификатом в DER-кодировке или PEM-формате.

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_ShowCrl** (context_t hCtx, mem_blk_t * crl)

Функция визуализации САС

Аргументы:

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

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_ShowCrlEx** (context_t hCtx, void * hWnd, mem_blk_t * crl)

Расширенная функция визуализации САС

Аргументы:

- **hCtx** (in) контекст библиотеки;

- **hWnd** (in) идентификатор родительского окна (см. описание в п. 1.11.1);
- **crl** (in) блок памяти с САС в DER-кодировке или PEM-формате.

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_ShowRequestEx** (context_t hCtx, void * hWnd, mem_blk_t * req)

Расширенная функция визуализации PKCS#10 запроса

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **hWnd** (in) идентификатор родительского окна (см. описание в п. 1.11.1);
- **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.

error_status_t VCERT1API **VCERT_ShowRevocationEx** (context_t hCtx, void * hWnd, mem_blk_t * rev)

Расширенная функция визуализации запроса на аннулирование сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **hWnd** (in) идентификатор родительского окна (см. описание в п. 1.11.1);
- **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:

- flag_t **flag**

Поле с маской (битовым ИЛИ) флагов формирования запроса PKCS#10

или запроса на аннулирование (должен быть установлен один и только один из флагов **FLAG_EXPORT_PKCS10REQUEST**, **FLAG_EXPORT_REVOKEREQUEST**):

- **FLAG_EXPORT_SHOWREQUESTUI** - автоматически визуализировать сформированный запрос PKCS#10 или запрос на аннулирование;
- **FLAG_EXPORT_PUREDERFORMAT** - экспортировать запрос PKCS#10 или запрос на аннулирование в DER-кодировке, без обертывания ЭП в формате CMS;
- **FLAG_EXPORT_PKCS10REQUEST** - выполнить формирование запроса PKCS#10 для проведения плановой смены рабочего сертификата пользователя;
- **FLAG_EXPORT_REVOKEREQUEST** - выполнить формирование запроса на аннулирование рабочего сертификата при компрометации закрытого ключа пользователя;
- **FLAG_EXPORT_GOST_R_34_10_12_256** - выполнить генерацию закрытого ключа для сертификата в квалифицированном формате по ГОСТ Р 34.10-2012 (256 бит);
- **FLAG_EXPORT_GOST_R_34_10_12_512** - выполнить генерацию закрытого ключа для сертификата в квалифицированном формате по ГОСТ Р 34.10-2012 (512 бит);
- **FLAG_EXPORT_CARRIER_GENERATION** - при генерации неизвлекаемого закрытого ключа на ФКН vdToken формировать этот ключ внутри ФКН с использованием внутреннего ДСЧ последнего.

– certid_t * **mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

Структура **import_param_t**

Параметры добавления объектов СУС в ПСП, ЛСП или ССС:

– flag_t **flag**

Поле с маской (побитовым ИЛИ) флагов 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**):

- **FLAG_IMPORT_DISPLAYOBJECTUI** - автоматически визуализировать импортируемые в ПСП, ЛСП или ССС объекты СУС;
- **FLAG_IMPORT_CERTDERPEMFORMAT** - импортировать сертификат в DER-кодировке или PEM-формате;
- **FLAG_IMPORT_CRLDERPEMFORMAT** - импортировать САС в DER-кодировке или PEM-формате;
- **FLAG_IMPORT_CERTWITHPRIVATE** - импортировать сертификат в DER-кодировке или PEM-формате и установить его в качестве рабочего сертификата пользователя;

- **FLAG_IMPORT_SIGNEDCARAUPDATE** - импортировать сертификаты и САС, содержащиеся в подписанном обновлении от ЦС и ЦР;
- **FLAG_IMPORT_DONOTVERIFYCHAIN** - импортировать сертификат или САС без построения и проверки цепочки сертификации;
- **FLAG_IMPORT_FORCEREMOTESTORE** - принудительно импортировать сертификат или САС в CCC.

– **certid_t * mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

1.14.2 Функции экспорта и импорта объектов СУС

error_status_t VCERT1API VCERT_ExportMem (context_t hCtx, export_param_t * pExportPara, mem_blk_t * oexp)

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

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pExportPara** (in) структура параметров формирования запроса PKCS#10/на аннулирование;
- **oexp** (out) блок данных с сформированным запросом в DER-кодировке (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API VCERT_ExportFile (context_t hCtx, export_param_t * pExportPara, const char * oexp)

Функция формирования файла с запросом PKCS#10/на аннулирование

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API VCERT_ExportToSystemStore (context_t hCtx, flag_t flag)

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

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_ImportMem** (context_t hCtx, import_param_t * pImportPara, mem_blk_t * iimp)

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

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_ImportFile** (context_t hCtx, import_param_t * pImportPara, const char * iimp)

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

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.15 Разбор и получение информации об объектах СУС

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

Функции также можно использовать для разбора дополнений (расширений) сертификатов из DER-кодировки. Для разбора дополнения (расширения) его необходимо найти в массиве дополнений (расширений) сертификата (см. описание в подразделе 1.7) по объектному идентификатору (OID), и передать блок памяти с закодированными данными дополнения в соответствующую функцию разбора:

- альтернативное имя владельца (OID **2.5.29.17**) - функция разбора **VCERT_ParseAltnameEx()**;

- альтернативное имя издателя (OID **2.5.29.18**) - функция разбора **VCERT_ParseAltnameEx()**;
- базовые ограничения сертификата (OID **2.5.29.19**) - функция разбора **VCERT_ParseBasicConstraintsEx()**;
- идентификатор ключа издателя (OID **2.5.29.35**) - функция разбора **VCERT_ParseKeyIdentifierEx()**.

1.15.1 Структуры разбора и получения информации об объектах СУС

Структура **othername_t**

Другое имя альтернативного имени:

- **string_t oid**

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

- **string_t name**

Строка с текстовыми данными другого имени.

Структура **altname_ex_t**

Другие имена альтернативного имени:

- **uint32_t 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:

- **uint32_t ca**

*Признак сертификата ЦС. Если значение **!= 0**, то сертификат является сертификатом ЦС.*

- **int32_t pathlen**

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

1.15.2 Функции разбора и получения информации об объектах СУС

error_status_t VCERT1API VCERT_ParseCert (**context_t hCtx**, **mem_blk_t * icert**, **certinfo_t info**, **certificate_t * ocert**)

Функция разбора и получения информации о сертификате

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **icert** (in) блок памяти с сертификатом в DER-кодировке или PEM-формате;
- **info** (in) требуемая возвращаемая информация о сертификате;
- **ocert** (out) структура разобранного сертификата (освобождается функцией **VCERT_FreeCert()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_ParseCrl** (context_t hCtx, mem_blk_t * icrl, crlinfo_t info, crl_t * ocrl)

Функция разбора и получения информации о САС

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **icrl** (in) блок памяти с САС в DER-кодировке или PEM-формате;
- **info** (in) требуемая возвращаемая информация о САС;
- **ocrl** (out) структура разобранного САС (освобождается функцией **VCERT_FreeCrl()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_ParseAltnameEx** (context_t hCtx, mem_blk_t * ialtname, altname_ex_t * oaltname)

Функция разбора и получения информации об альтернативном имени

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **ialtname** (in) блок памяти с данными дополнения альтернативного имени в DER-кодировке;
- **oaltname** (out) структура разобранного альтернативного имени (освобождается функцией **VCERT_FreeAltnameEx()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_ParseBasicConstraintsEx** (context_t hCtx, mem_blk_t * ibc, basic_constraints_ex_t * obc)

Функция разбора и получения информации о базовых ограничениях сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_ParseKeyIdentifierEx** (context_t hCtx, mem_blk_t * ikid, mem_blk_t * okid)

Функция разбора и получения информации об идентификаторе ключа издателя

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **ikid** (in) блок памяти с данными дополнения идентификатора ключа издателя в DER-кодировке;
- **okid** (out) блок памяти с разобранным идентификатором ключа издателя (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

1.16 Построение и проверка цепочек объектов СУС

Доверие любому сертификату и САС определяется на основании результата построения и проверки его цепочки сертификации. Начальным элементом цепочки сертификации является сертификат доверенного (корневого) ЦС, хранящийся в защищенном ПСП. Библиотека в своем составе содержит функции для построения и проверки (верификации) цепочек сертификатов и САС.

Процедура верификации цепочки сертификации проверяет связанность между именем **владельца** сертификата и его открытым ключом. Процедура верификации цепочки подразумевает, что все действительные цепочки начинаются с сертификата, изданного одним из **доверенных (корневых)** ЦС. Алгоритм верификации требует открытый ключ, имена **издателя** и **владельца** сертификата, срок действия сертификата и его закрытого ключа, а также САС для каждого ЦС (САС необходим, даже если он не содержит аннулированных сертификатов).

Под доверенным понимается ЦС, открытый ключ которого содержится в самоподписанном сертификате. Такое ограничение упрощает процедуру верификации, хотя наличие самоподписанного сертификата и его криптографическая проверка не обеспечивают безопасности. Для обеспечения доверия открытому ключу такого сертификата должны быть применены специальные способы его распространения и хранения, так как на данном открытом ключе проверяются все остальные сертификаты.

Цепочка сертификации представляет собой последовательность из **n** сертификатов, в которой:

- для **x** в $\{1, (n-1)\}$, **владелец** сертификата **x** есть **издатель** сертификата **x+1**;
- сертификат **x=1** есть самоподписанный сертификат;
- сертификат **x=n** есть проверяемый сертификат.

Одновременно в цепочке сертификации присутствует последовательность из **n-1** САС, в которой:

- для САС **x** в $\{1, (n-1)\}$, **издатель** сертификата **x** есть **издатель** САС **x**;
- САС **x=1** есть САС, изданный **владельцем** самоподписанного сертификата;
- САС **x=(n-1)** есть САС, изданный **издателем** проверяемого сертификата.

При построении цепочки сертификат доверенного (корневого) ЦС ищется исключительно в ПСП. Сертификаты промежуточных ЦС ищутся в ЛСП и в точках AIA типа **id-ad-calssuers** с объектным идентификатором (OID) 1.3.6.1.5.5.7.48.2 (дополнение **authorityInfoAccess**). САС ищутся в ЛСП и в точках CDP (дополнение **crlDistributionPoints**).

После построения цепочки сертификации выполняется криптографическая проверка сертификатов и САС, проверка сроков действия сертификатов и САС, проверка действительности закрытых ключей на момент вычисления ЭП, проверка сертификатов на аннулирование. Если сертификат промежуточного ЦС цепочки аннулирован в САС вышестоящего ЦС, то все сертификаты, изданные с использованием данного сертификата промежуточного ЦС, считаются недействительными.

Схема верификации цепочки сертификации приведена ниже (Рисунок 1). На схеме рассмотрены два случая:

– первый случай - верификация цепочки сертификации от проверяемого сертификата до сертификата доверенного (корневого) ЦС. Определение САС выполняется для каждого сертификата ЦС по имени издателя САС;

– второй случай - (нижняя часть схемы) отличается тем, что в системе одновременно присутствуют два сертификата доверенного (корневого) ЦС:

- первый - замыкающий цепочку от проверяемого сертификата;
- второй - новый сертификат доверенного (корневого) ЦС. После формирования нового сертификата доверенного (корневого) ЦС в системе появляется САС, изданный на новом закрытом ключе ЦС, который заменяет предыдущий САС и содержит все ранее аннулированные сертификаты. Для проверки нового САС требуется однозначное определение сертификата его издателя, т.е. однозначное определение открытого ключа для проверки ЭП данного САС. Сертификат издателя САС однозначно определяется по идентификатору ключа издателя (дополнение **authorityKeyIdentifier**).

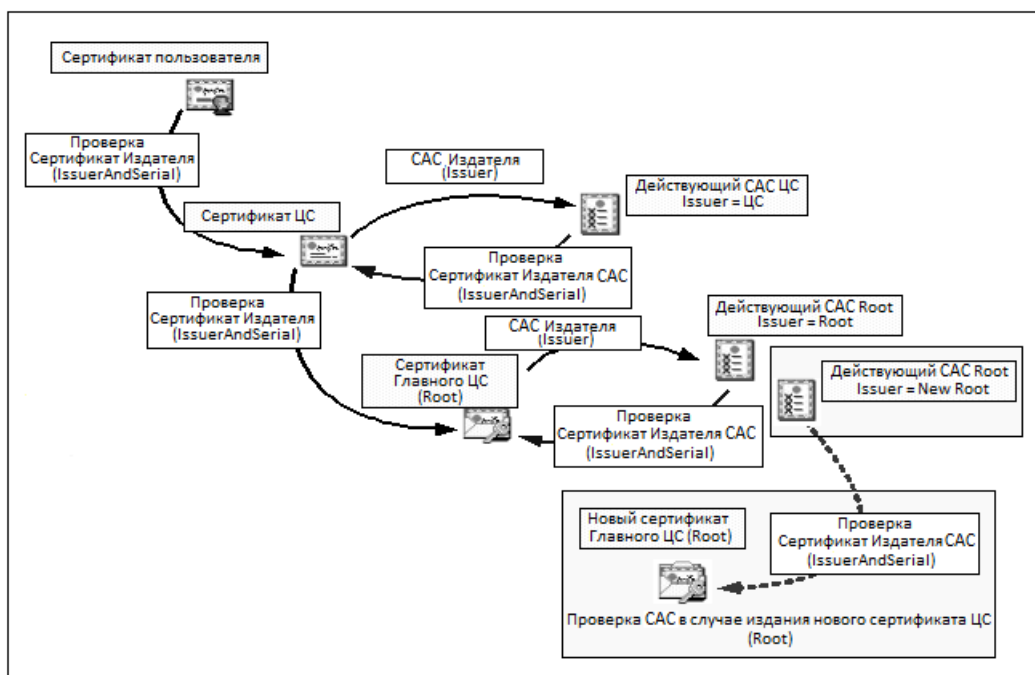


Рисунок 1 – Верификация цепочки сертификации

1.16.1 Структуры построения и проверки цепочек объектов СУС

Структура `verify_policy_param_t`

Структура параметров построения и проверки цепочки сертификата или CAC:

– `uint32_t size`

Поле с размером структуры в байтах, значение должно быть равно `sizeof(verify_policy_param_t)`.

– `flag_t flag`

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

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

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

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

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

– `certid_t * mycert`

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

– `date_t check_time`

Поле с моментом времени, на который строится и проверяется цепочка сер-

тификата. Если значение равно **0**, то проверка выполняется на текущий момент времени. Данное поле следует заполнять в тех случаях, когда (согласно Статье 11 №63-ФЗ) имеется достоверная информация о моменте подписания электронного документа - например, данные о моменте подписания, взятые из успешно проверенного штампа времени (см. описание в разделе 1.26).

– `keyusage_t` **keyusage**

Поле с маской (битовым 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.

– `uint32_t` **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.

– `uint32_t` **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.

1.16.2 Функции построения и проверки цепочек объектов СУС

`error_status_t` VCERT1API **VCERT_VerifyCert** (`context_t` hCtx, `verify_param_t *` pVerifyPara, `mem_blk_t *` icert, `certificate_t *` ocert)

Функция построения и проверки цепочки сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pVerifyPara** (in) структура параметров проверки сертификата (см. описание в п. 1.21.1). 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 (освобождается функцией **VCERT_FreeCert()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

`error_status_t` VCERT1API **VCERT_VerifyCertificatePolicy** (`context_t` hCtx, `verify_policy_param_t *` pPolicyPara, `mem_blk_t *` icert)

Расширенная функция построения и проверки цепочки сертификата**Аргументы:**

- **hCtx** (in) контекст библиотеки;
- **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.

```
error_status_t VCERT1API VCERT_VerifyCrlPolicy (context_t hCtx, verify_policy_param_t * pPolicyPara, mem_blk_t * icrl)
```

Расширенная функция построения и проверки цепочки САС**Аргументы:**

- **hCtx** (in) контекст библиотеки;
- **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.

1.17 Вычисление хэш-значений

1.17.1 Функции вычисления хэш-значений

```
error_status_t VCERT1API VCERT_BlzHashMem (context_t hCtx, string_t algorithm, mem_blk_t * data, mem_blk_t * hash)
```

Функция блочного вычисления хэш-значения блока памяти**Аргументы:**

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_BlkHashFile** (context_t hCtx, string_t algorithm, const char * data, mem_blk_t * hash)

Функция блочного вычисления хэш-значения файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_StrHashInitMem** (context_t hCtx, string_t algorithm, strhash_t * hStr)

Функция инициализации потокового вычисления хэш-значения блоков памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_StrHashUpdateMem** (context_t hCtx, strhash_t * hStr, mem_blk_t * data)

Функция продолжения потокового вычисления хэш-значения блоков памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_StrHashFinalMem** (context_t hCtx, strhash_t * hStr, mem_blk_t * hash)

Функция финализации потокового вычисления хэш-значения блоков памяти

Аргументы:

– **hCtx** (in) контекст библиотеки;
 – **hStr** (in) контекст выполнения потоковой операции;
 – **hash** (out) блок памяти для вычисленного хэш-значения (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_StrHashFile** (context_t hCtx, string_t algorithm, const char * data, mem_blk_t * hash)

Функция потокового вычисления хэш-значения файла

Аргументы:

– **hCtx** (in) контекст библиотеки;
 – **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

1.18 Вычисление и проверка ЭП хэш-значений

1.18.1 Функции вычисления и проверки ЭП хэш-значений

error_status_t VCERT1API **VCERT_SignHashMem** (context_t hCtx, certid_t * mycert, mem_blk_t * hash, mem_blk_t * sign)

Функция вычисления ЭП хэш-значения

Аргументы:

– **hCtx** (in) контекст библиотеки;

- **mycert** (in) структура идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки). Вычисление ЭП выполняется на ключе ЭП, соответствующем указанному сертификату;
- **hash** (in) блок памяти с хэш-значением, вычисленным по ГОСТ Р 34.11-2012 (см. описание функций в п. 1.17.1);
- **sign** (out) блок памяти для вычисленной ЭП хэш-значения (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_VerifyHashMem** (context_t hCtx, certid_t * mycert, mem_blk_t * sender, mem_blk_t * hash, mem_blk_t * sign)

Функция проверки ЭП хэш-значения

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **mycert** (in) структура идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки);
- **sender** (in) блок памяти с сертификатом, на котором необходимо проверять ЭП (в DER-кодировке или PEM-формате);
- **hash** (in) блок памяти с хэш-значением, вычисленным по ГОСТ Р 34.11-2012 (см. описание функций в п. 1.17.1);
- **sign** (in) блок памяти с проверяемой ЭП хэш-значения.

Возвращаемые значения:

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

1.19 Поиск и перечисление объектов СУС

Функции поиска объектов позволяют находить сертификаты пользователей в доступных справочниках по заданному шаблону сертификата. Шаблон представляет собой структуру сертификата **certificate_t**, пустую или заполненную теми данными, по которым следует выполнять операцию поиска. Каждый считанный из справочников сертификат пользователя проверяется на соответствие заполненным данным шаблона, и при успешной проверке включается в структуру результата поиска, содержащую найденные сертификаты.

Если в шаблоне сертификата заполнено поле с блоком памяти, содержащим сертификат в DER-кодировке, то фактический поиск в справочниках не производится, а сертификат проверяется на соответствие заполненным данным шаблона.

Поиск выполняется в доступных справочниках в такой последовательности: кэш библиотеки, ЛСП, ССС. Поиск сертификата выполняется по уникальному критерию, если в шаблоне сертификата заполнены любые из нижеследующих комбинаций данных:

- пара X.500-имени издателя и серийного номера сертификата;

- идентификатор ключа владельца сертификата;
- идентификатор закрытого ключа сертификата;
- хэш-значение пары X.500-имени издателя и серийного номера сертификата.

Поиск по уникальному критерию выполняется до первого найденного сертификата, после чего операция поиска сертификатов в доступных справочниках завершается, а найденный сертификат проверяется на соответствие заполненным данным шаблона.

Поиск по паре X.500-имени издателя и серийного номера сертификата, а также по хэш-значению такой пары, выполняется быстро во всех справочниках, поскольку в них сформирован требуемый индекс для повышения производительности.

Поиск по идентификатору ключа владельца или идентификатору закрытого ключа в ЛСП, расположенном в БД GDBM, выполняется медленно, поскольку требует перебора объектов хранилища. В других справочниках такой поиск выполняется быстро, поскольку в них сформирован требуемый индекс (для кэша библиотеки необходимо выполнить настройку - см. описание соответствующих параметров **Pki1CacheSbKeyIdIndex** и **Pki1CacheKeyIdIndex** в таблице 1).

Поиск не по уникальному критерию выполняется во всех доступных справочниках, и все найденные сертификаты проверяются на соответствие заполненным данным шаблона. В общем случае подобный поиск выполняется медленно, поскольку в справочниках отсутствуют необходимые индексы. Исключениями являются следующие типы поиска:

- поиск по полному X.500-имени владельца или серийному номеру сертификата, а также по адресу электронной почты RFC 822 в ЛСП, расположенном в БД ODBC (поиск по части X.500-имени владельца выполняется медленно);
- поиск по полному X.500-имени владельца сертификата в ССС (поиск по части X.500-имени владельца выполняется медленно).

Функции перечисления объектов позволяют последовательно вычитывать объекты требуемого типа - сертификаты, САС - из указанного справочника - ПСП, ЛСП, ССС, или справочника, идентифицируемого посредством URI.

Процедуру перечисления объектов необходимо продолжать и завершать в том же потоке, в котором данная процедура была начата. В рамках одного процесса разрешается выполнять только одну процедуру перечисления - т.е. начинать следующую процедуру перечисления объектов можно только после завершения предыдущей.

Во время перечисления объектов ЛСП, расположенного в БД GDBM, разрешено вызывать только функции разбора объектов СУС. Это ограничение связано с тем, что при открытом контексте перечисления объектов запрещен доступ к ЛСП из других функций библиотеки.

1.19.1 Структуры поиска сертификатов

Структура **find_param_t**

Структура параметров поиска сертификатов по заданному шаблону:

– flag_t **flag**

Поле с маской (побитовым ИЛИ) флагов поиска сертификатов:

- **FLAG_FIND_CERTWITHPRIVATE** - выполнять поиск только тех сертификатов, для которых есть закрытый ключ (рабочих сертификатов);
- **FLAG_FIND_USEREMOTESEARCH** - выполнять поиск сертификатов также и в CCC (по умолчанию такой поиск не выполняется);
- **FLAG_FIND_SHOWUISELECTOR** - показать пользователю диалоговый интерфейс, отображающий найденные сертификаты;
- **FLAG_FIND_DONOTCHECKTIMES** - не проверять сроки действия найденных сертификатов, а также сертификатов ЦС и САС из их цепочек;
- **FLAG_FIND_DONOTVERIFYCHAIN** - не строить и не проверять цепочки найденных сертификатов;
- **FLAG_FIND_ADDREMOTETOLOCAL** - автоматически добавлять сертификат в ЛСП в случае нахождения его в CCC по уникальному критерию;
- **FLAG_FIND_DONOTCHECKKEYTIME** - не проверять сроки действия закрытых ключей найденных сертификатов;
- **FLAG_FIND_SUBJECTISPARTIAL** - выполнять поиск сертификатов по части X.500-имени владельца (установка данного флага показывает, что в поле subject шаблона сертификата находится часть X.500-имени владельца сертификата - например, "OU=7395399001"). Такой поиск поддерживается только для кэша библиотеки, для ЛСП, расположенного в БД ODBC, а также для CCC - в этом случае поиск выполняется по атрибуту **vdSubjName**;
- **FLAG_FIND_IGNORESTORECACHE** - не выполнять поиск в кэше библиотеки (по умолчанию такой поиск выполняется);
- **FLAG_FIND_IGNORESTORELOCAL** - не выполнять поиск в ЛСП (по умолчанию такой поиск выполняется);
- **FLAG_FIND_SUBJECTATTRIBUTE** - выполнять поиск по полностью заданному X.500-имени владельца сертификата в CCC по атрибуту **vdSubjName**, а не по атрибуту **distinguishedName**. Использование данного флага позволяет находить сертификаты в CCC независимо от имен контейнеров, в которых они фактически расположены.

– certid_t * **mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

– certificate_t **certTemplate**

Поле со структурой шаблона сертификата для выполнения поиска.

– certinfo_t **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:

– uint32_t **num**

Количество элементов массива найденных сертификатов.

– certificate_t * **certs**

Поле с массивом найденных сертификатов.

1.19.2 Функции поиска сертификатов

error_status_t VCERT1API **VCERT_FindCert** (context_t hCtx, find_param_t * pFindPara, find_result_t * pFindResult)

Функция поиска сертификатов по заданному шаблону

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **pFindPara** (in) структура параметров поиска сертификатов по заданному шаблону;

– **pFindResult** (out) структура результата поиска сертификатов по заданному шаблону (освобождается функцией **VCERT_FreeFindResult()** - см. описание в п. 1.31.1).

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.

1.19.3 Функции перечисления объектов СУС

error_status_t VCERT1API **VCERT_EnumStoreObjects** (context_t hCtx, certid_t * mycert, enuobj_t * hEnu, mem_blk_t * object, flag_t flag)

Функция перечисления объектов справочников

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **mycert** (in) структура идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки);

– **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);

– **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**):

• **FLAG_ENUM_STORE_OBJECTS_STORE_PERS** - выполнять перечисление объектов ПСП (данный флаг нельзя использовать с минимальным контекстом библиотеки);

• **FLAG_ENUM_STORE_OBJECTS_STORE_LOCL** - выполнять перечисление объектов ЛСП (данный флаг нельзя использовать с минимальным контекстом библиотеки);

- **FLAG_ENUM_STORE_OBJECTS_STORE_LDAP** - выполнять перечисление объектов CCC (данный флаг нельзя использовать с минимальным контекстом библиотеки);
- **FLAG_ENUM_STORE_OBJECTS_OBJECT_CER** - выполнять перечисление сертификатов;
- **FLAG_ENUM_STORE_OBJECTS_OBJECT_CRL** - выполнять перечисление CAC;
- **FLAG_ENUM_STORE_OBJECTS_CLOSE_ENUM** - прекратить процесс перечисления и освободить указанный контекст. Данный флаг должен быть добавлен через побитовое ИЛИ к маске флагов, с которыми выполнялось перечисление объектов.

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**.

error_status_t VCERT1API **VCERT_EnumStoreObjectsEx** (context_t hCtx, certid_t* mycert, const char* suri, string_t lqry, enuobj_t* hEnu, mem_blk_t* object, flag_t flag)

Расширенная функция перечисления объектов справочников

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **mycert** (in) структура идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки);
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **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**);

- **FLAG_ENUM_STORE_OBJECTS_STORE_PERS** - выполнять перечисление объектов ПСП (данный флаг нельзя использовать с минимальным контекстом библиотеки);
- **FLAG_ENUM_STORE_OBJECTS_STORE_LOCL** - выполнять перечисле-

ние объектов ЛСП (данный флаг нельзя использовать с минимальным контекстом библиотеки);

- **FLAG_ENUM_STORE_OBJECTS_STORE_LDAP** - выполнять перечисление объектов ССС (данный флаг нельзя использовать с минимальным контекстом библиотеки);

- **FLAG_ENUM_STORE_OBJECTS_STORE_SURI** - выполнять перечисление объектов справочника по заданному URI (данный флаг можно использовать с минимальным контекстом библиотеки);

- **FLAG_ENUM_STORE_OBJECTS_OBJECT_CER** - выполнять перечисление сертификатов;

- **FLAG_ENUM_STORE_OBJECTS_OBJECT_CRL** - выполнять перечисление САС;

- **FLAG_ENUM_STORE_OBJECTS_CLOSE_ENUM** - прекратить процесс перечисления и освободить указанный контекст. Данный флаг должен быть добавлен через побитовое ИЛИ к маске флагов, с которыми выполнялось перечисление объектов.

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**.

1.20 Вычисление ЭП CMS-сообщений

Библиотека поддерживает следующие типы подписанных CMS-сообщений:

- CMS-сообщение с совмещенной(ыми) (присоединенной(ыми), или Attached) ЭП. В таком CMS-сообщении присутствуют и подписанные данные, и ЭП этих данных;

- CMS-сообщение с отделенной(ыми) (отсоединенной(ыми), или Detached) ЭП. В таком CMS-сообщении присутствуют только ЭП подписанных данных.

Формат подписанных CMS-сообщений - CMS Signed - является универсальным за счет реализации с использованием стандарта ASN.1. Идентификация сертификата(ов) подписанта(ов) осуществляется либо посредством пары Имя издателя/Серийный номер, либо посредством использования данных дополнения "Идентификатор ключа владельца". Формат CMS Signed позволяет добавлять в CMS-сообщение сертификаты, САС, а также добавлять в каждую ЭП дополнительные аутентифицируемые и неаутентифицируемые атрибуты.

Перед вычислением ЭП CMS-сообщения функции подписания предварительно производят:

- проверку действительности рабочего сертификата на текущий момент времени;

- проверку действительности закрытого ключа на текущий момент времени;

- проверку наличия в рабочем сертификате разрешенной области использования открытого ключа **KEYUSAGE_DIGITAL_SIGNATURE** (т.е. проверку возможности вычисления ЭП на закрытом ключе).

При подписании в блочном режиме, а также при подписании в потоковом режиме с отделенной(ыми) ЭП подписанное CMS-сообщение имеет ASN.1 кодировку определенной длины (ASN.1 definite-length encoding). При подписании в потоковом режиме с совмещенной(ыми) ЭП подписанное CMS-сообщение имеет ASN.1 кодировку неопределенной длины (ASN.1 indefinite-length encoding).

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 отделенной(ыми) ЭП, ни тип ASN.1 кодировки - определенной или неопределенной длины.

1.20.1 Структуры вычисления ЭП CMS-сообщений

Структура `sign_param_t`

Структура параметров вычисления ЭП CMS-сообщений:

– `flag_t flag`

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

- **FLAG_CMS_SIGN_ADDSIGNER** - добавлять сертификат подписанта - фактически, рабочий сертификат - в CMS-сообщение;

- **FLAG_CMS_SIGN_SUBJKEYID** - использовать для идентификации сертификата подписанта данные дополнения "Идентификатор ключа владельца". Если данный флаг не установлен, то для идентификации используется пара Имя издателя/Серийный номер;

- **FLAG_CMS_SIGN_ENVELOPE** - не добавлять ЭП к подписанному CMS-сообщению, а обернуть ЭП существующее CMS-сообщение целиком. Данный флаг разрешено использовать только при формировании совмещенной ЭП;

- **FLAG_CMS_SIGN_CADESBES** - добавить в ЭП аутентифицируемые атрибуты в соответствии со спецификацией **CAdES-BES**. Установка данного флага приводит к добавлению дополнительного аутентифицируемого атрибута **ESS signing-certificate**.

– `certid_t * mycert`

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

1.20.2 Функции блочного вычисления совмещенной ЭП CMS-сообщений

`error_status_t VCERT1API VCERT_CmsBlkAttSignMem (context_t hCtx, sign_param_t * pSignPara, mem_blk_t * data, mem_blk_t * ocms)`

Функция блочного вычисления совмещенной ЭП блока памяти

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsBlkAttSignFile** (context_t hCtx, sign_param_t * pSignPara, const char * data, const char * ocms)

Функция блочного вычисления совмещенной ЭП файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.20.3 Функции потокового вычисления совмещенной ЭП CMS-сообщений

error_status_t VCERT1API **VCERT_CmsStrAttSignInitMem** (context_t hCtx, sign_param_t * pSignPara, mem_blk_t * data, strcms_t * hStr)

Функция инициализации потокового вычисления совмещенной ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_CmsStrAttSignUpdateMem** (context_t hCtx, sign_param_t * pSignPara, strcms_t * hStr, mem_blk_t * data, mem_blk_t * ocms)

Функция продолжения потокового вычисления совмещенной ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrAttSignFinalMem** (context_t hCtx, sign_param_t * pSignPara, strcms_t * hStr, mem_blk_t * ocms)

Функция финализации потокового вычисления совмещенной ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrAttSignFile** (context_t hCtx, sign_param_t * pSignPara, const char * data, const char * ocms)

Функция потокового вычисления совмещенной ЭП файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.20.4 Функции блочного вычисления отдельной ЭП CMS-сообщений

error_status_t VCERT1API **VCERT_CmsBlkDetSignMem** (context_t hCtx, sign_param_t * pSignPara, mem_blk_t * data, mem_blk_t * icms, mem_blk_t * ocms)

Функция блочного вычисления отдельной ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsBlkDetSignFile** (context_t hCtx, sign_param_t * pSignPara, const char * data, const char * icms, const char * ocms)

Функция блочного вычисления отделенной ЭП файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.20.5 Функции потокового вычисления отделенной ЭП CMS-сообщений

error_status_t VCERT1API **VCERT_CmsStrDetSignInitMem** (context_t hCtx, sign_param_t * pSignPara, mem_blk_t * icms, strcms_t * hStr)

Функция инициализации потокового вычисления отделенной ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_CmsStrDetSignUpdateMem** (context_t hCtx, sign_param_t * pSignPara, strcms_t * hStr, mem_blk_t * data)

Функция продолжения потокового вычисления отделенной ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;

– **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.

error_status_t VCERT1API **VCERT_CmsStrDetSignFinalMem** (context_t hCtx, sign_param_t * pSignPara, strcms_t * hStr, mem_blk_t * ocms)

Функция финализации потокового вычисления отделенной ЭП блока памяти

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrDetSignFile** (context_t hCtx, sign_param_t * pSignPara, const char * data, const char * icms, const char * ocms)

Функция потокового вычисления отделенной ЭП файла

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **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.

1.21 Проверка ЭП CMS-сообщений

Для использования в полной мере возможностей, заложенных в СУС, в процессе проверки ЭП CMS-сообщений от прикладного ПО требуются следующие исходные данные:

– подписанное CMS-сообщение;

– набор приемлемых для прикладного ПО регламентов использования сер-

тификата. Данный набор представляет собой список идентификаторов, по сути аналогичных идентификаторам регламентов использования сертификата, которые определяют приемлемость сертификата(ов) подписанта(ов) при проверке ЭП подписанного CMS-сообщения;

– набор приемлемых для прикладного ПО расширенных использований открытого ключа. Данный набор представляет собой список идентификаторов, по сути аналогичных идентификаторам расширенных использований открытого ключа, которые определяют приемлемость сертификата(ов) подписанта(ов) при проверке ЭП подписанного CMS-сообщения.

Примечание - Требование предоставления прикладным ПО наборов приемлемых регламентов использования сертификата и расширенных использований открытого ключа не является обязательным, однако их использование позволяет в полной мере реализовать возможности библиотеки в части обеспечения политики безопасности для конкретного прикладного ПО.

Общий алгоритм проверки ЭП CMS-сообщения может быть представлен в следующем виде:

- анализ CMS-сообщения с целью определения количества ЭП в нем и поиска сертификата(ов) подписанта(ов);
- собственно проверка ЭП CMS-сообщения, после которой(ых) выполняется построение и проверка цепочек сертификата(ов) подписанта(ов);
- анализ сертификата(ов) подписанта(ов) на наличие требуемых регламентов использования сертификата и расширенных использований открытого ключа.

При проверке ЭП CMS-сообщения для каждого сертификата подписанта производятся следующие действия:

- выполняется проверка сроков действия всех сертификатов и всех САС, составляющих цепочку сертификата подписанта;
- выполняется проверка действия закрытых ключей сертификатов цепочки на моменты вычисления соответствующих ЭП;
- выполняется криптографическая проверка всех сертификатов и САС цепочки сертификата подписанта;
- при заданных требуемых регламентах использования сертификата и расширенных использованиях открытого ключа производится проверка на их наличие в соответствующих дополнениях сертификата подписанта (для успешной проверки необходимо, чтобы все требуемые идентификаторы были представлены в сертификате);
- если сертификат подписанта присутствует в САС, функция возвращает:
 - при неустановленном флаге FLAG_CMS_VERIFY_USEREVOCATIONTIME
 - код успешной проверки, если время аннулирования сертификата еще не наступило;
 - код ошибки, если время аннулирования сертификата уже наступило;
 - при установленном флаге FLAG_CMS_VERIFY_USEREVOCATIONTIME

- код успешной проверки, если документ был подписан до аннулирования сертификата;
- код ошибки, если документ был подписан после аннулирования сертификата.

Для корректной обработки результатов проверки ЭП CMS-сообщения прикладное ПО должно анализировать результат проверки каждой ЭП индивидуально.

Результаты проверки ЭП на аннулированном сертификате приведены в таблицах 2 и 3. Для обозначения времени вычисления ЭП используется $T_{\text{вычисления}}$, времени проверки ЭП - $T_{\text{проверки}}$, а времени аннулирования сертификата - $T_{\text{аннулирования}}$.

При вычислении и проверке ЭП времена $T_{\text{вычисления}}$ и $T_{\text{проверки}}$ определяются по системному времени вычислительной машины, на которой выполняется данная конкретная операция.

Таблица 2 – Результаты проверки ЭП на аннулированном сертификате (без использования флага **FLAG_CMS_VERIFY_USEREVOCATIONTIME**)

Условия проверки ЭП	Результат проверки ЭП
$T_{\text{вычисления}} < T_{\text{проверки}} < T_{\text{аннулирования}}$	Проверка выполнена успешно
$T_{\text{проверки}} \leq T_{\text{вычисления}} < T_{\text{аннулирования}}$	Проверка выполнена успешно
$T_{\text{вычисления}} < T_{\text{аннулирования}} \leq T_{\text{проверки}}$	Сертификат аннулирован
$T_{\text{проверки}} < T_{\text{аннулирования}} \leq T_{\text{вычисления}}$	Сертификат аннулирован
$T_{\text{аннулирования}} < T_{\text{проверки}} \leq T_{\text{вычисления}}$	Сертификат аннулирован
$T_{\text{аннулирования}} \leq T_{\text{вычисления}} \leq T_{\text{проверки}}$	Сертификат аннулирован

Таблица 3 – Результаты проверки ЭП на аннулированном сертификате (с использованием флага **FLAG_CMS_VERIFY_USEREVOCATIONTIME**)

Условия проверки ЭП	Результат проверки ЭП
$T_{\text{вычисления}} < T_{\text{проверки}} < T_{\text{аннулирования}}$	Проверка выполнена успешно
$T_{\text{проверки}} \leq T_{\text{вычисления}} < T_{\text{аннулирования}}$	Проверка выполнена успешно
$T_{\text{вычисления}} < T_{\text{аннулирования}} \leq T_{\text{проверки}}$	Проверка выполнена успешно
$T_{\text{проверки}} < T_{\text{аннулирования}} \leq T_{\text{вычисления}}$	Сертификат аннулирован
$T_{\text{аннулирования}} < T_{\text{проверки}} \leq T_{\text{вычисления}}$	Сертификат аннулирован
$T_{\text{аннулирования}} \leq T_{\text{вычисления}} \leq T_{\text{проверки}}$	Сертификат аннулирован

1.21.1 Структуры проверки ЭП CMS-сообщений

Структура `verify_param_t`

Структура параметров проверки ЭП CMS-сообщений:

– flag_t **flag**

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

- **FLAG_CMS_VERIFY_DELETESIGNATURES** - в случае успешной проверки удалять ЭП в количестве, равном значению поля **nSignToDelete**, начиная с конца CMS-сообщения;
- **FLAG_CMS_VERIFY_SIGNERKEYUSAGES** - проверять наличие в сертификатах подписантов разрешенных областей использования открытого ключа, указанных в поле **keyUsage**;
- **FLAG_CMS_VERIFY_SIGNERPOLICIES** - проверять наличие в сертификатах подписантов регламентов использования сертификата, указанных в полях **policy_num** и **policies**;
- **FLAG_CMS_VERIFY_SIGNEREXTKEYUSAGES** - проверять наличие в сертификатах подписантов расширенных областей использования открытого ключа, указанных в полях **extkeyusage_num** и **extKeyUsage**;
- **FLAG_CMS_VERIFY_MINIMUMSIGNATURES** - проверять, что количество ЭП в CMS-сообщении больше или равно значению поля **minsigs**;
- **FLAG_CMS_VERIFY_DONOTCHECKTIMES** - не проверять сроки действия сертификатов подписантов, а также сертификатов ЦС и САС из их цепочек;
- **FLAG_CMS_VERIFY_DONOTADDSIGNER** - не добавлять автоматически сертификаты подписантов, найденные произвольным способом, в ЛСП;
- **FLAG_CMS_VERIFY_IGNOREATTACHEDSIGNER** - не искать сертификаты подписантов среди сертификатов и САС, прикрепленных к CMS-сообщению;
- **FLAG_CMS_VERIFY_USEREVOCATIONTIME** - учитывать времена вычисления ЭП и аннулирования сертификатов подписантов при построении и проверке их цепочек;
- **FLAG_CMS_VERIFY_DONOTADDATTACHEDSIGNER** - не добавлять автоматически сертификаты подписантов, прикрепленные к CMS-сообщению, в ЛСП;
- **FLAG_CMS_VERIFY_REQUIREATTACHEDSIGNER** - требовать наличия сертификатов подписантов среди сертификатов и САС, прикрепленных к CMS-сообщению;
- **FLAG_CMS_VERIFY_USEATTACHEDCHAIN** - использовать сертификаты промежуточных ЦС и САС, прикрепленные к CMS-сообщению, при построении и проверке цепочек сертификатов подписантов;
- **FLAG_CMS_VERIFY_REQUIREATTACHEDCHAIN** - требовать наличия сертификатов промежуточных ЦС и САС, прикрепленных к CMS-сообщению. Установка данного флага приводит к тому, что все сертификаты промежуточных ЦС и САС, необходимые для построения и проверки цепочек сертификатов подписантов, ищутся среди тех, что прикреплены к CMS сообщению.

– certid_t * **mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (*NULL* при использовании локальной библиотеки).

– `keyusage_t` **keyUsage**

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

– `uint32_t` **policy_num**

Количество элементов массива регламентов использования сертификата.

– `policy_t *` **policies**

Поле с массивом регламентов использования сертификата.

– `uint32_t` **extkeyusage_num**

Количество элементов массива расширенных областей использования открытого ключа.

– `extkeyusage_t *` **extKeyUsage**

Поле с массивом расширенных областей использования открытого ключа.

– `int32_t` **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**.

– `uint32_t` **minsigns**

Поле с минимально допустимым количеством ЭП в CMS-сообщении.

– `certinfo_t` **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.

– `date_t` **time**

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 с указателем на структуру 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:

– `uint32_t` **sign_num**

Количество элементов массива результатов проверки ЭП подписанного CMS-сообщения.

– `sign_status_t *` **signs**

Поле с массивом результатов проверки ЭП подписанного CMS-сообщения.

1.21.2 Функции блочной проверки совмещенных ЭП CMS-сообщений

error_status_t VCERT1API **VCERT_CmsBlkAttVerifyMem** (context_t hCtx, verify_param_t * pVerifyPara, mem_blk_t * icms, mem_blk_t * data, verify_result_t * pVerifyResult)

Функция блочной проверки совмещенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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** (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения (освобождается функцией **VCERT_FreeVerifyResult()** - см. описание в п. 1.31.1).

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**).

error_status_t VCERT1API **VCERT_CmsBlkAttVerifyFile** (context_t hCtx, verify_param_t * pVerifyPara, const char * icms, const char * data, verify_result_t * pVerifyResult)

Функция блочной проверки совмещенных ЭП файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeVerifyResult()** - см. описание в п. 1.31.1).

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**).

1.21.3 Функции потоковой проверки совмещенных ЭП CMS-сообщений

error_status_t VCERT1API VCERT_CmsStrAttVerifyInitMem (context_t hCtx, verify_param_t * pVerifyPara, mem_blk_t * icms, strcms_t * hStr)

Функция инициализации потоковой проверки совмещенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API VCERT_CmsStrAttVerifyUpdateMem (context_t hCtx, verify_param_t * pVerifyPara, strcms_t * hStr, mem_blk_t * icms, mem_blk_t * data)

Функция продолжения потоковой проверки совмещенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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** (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API VCERT_CmsStrAttVerifyFinalMem (context_t hCtx, verify_param_t * pVerifyPara, strcms_t * hStr, mem_blk_t * data, verify_result_t * pVerifyResult)

Функция финализации потоковой проверки совмещенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pVerifyPara** (in) структура параметров проверки ЭП CMS-сообщений (должна быть одинакова для функций инициализации, продолжения и финализации);

зации);

- **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** (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения (освобождается функцией **VCERT_FreeVerifyResult()** - см. описание в п. 1.31.1).

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**).

error_status_t VCERT1API **VCERT_CmsStrAttVerifyFile** (context_t hCtx, verify_param_t * pVerifyPara, const char * icms, const char * data, verify_result_t * pVerifyResult)

Функция потоковой проверки совмещенных ЭП файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeVerifyResult()** - см. описание в п. 1.31.1).

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**).

1.21.4 Функции блочной проверки отделенных ЭП CMS-сообщений

error_status_t VCERT1API **VCERT_CmsBlkDetVerifyMem** (context_t hCtx, 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:

- **hCtx** (in) контекст библиотеки;

- **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** (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения (освобождается функцией **VCERT_FreeVerifyResult()** - см. описание в п. 1.31.1).

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**).

error_status_t VCERT1API **VCERT_CmsBlkDetVerifyFile** (context_t hCtx, verify_param_t * pVerifyPara, const char * data, const char * icms, const char * ocms, verify_result_t * pVerifyResult)

Функция блочной проверки отделенных ЭП файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeVerifyResult()** - см. описание в п. 1.31.1).

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**).

1.21.5 Функции потоковой проверки отделенных ЭП CMS-сообщений

`error_status_t VCERT1API VCERT_CmsStrDetVerifyInitMem` (`context_t hCtx`, `verify_param_t * pVerifyPara`, `mem_blk_t * icms`, `strcms_t * hStr`)

Функция инициализации потоковой проверки отделенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

`error_status_t VCERT1API VCERT_CmsStrDetVerifyUpdateMem` (`context_t hCtx`, `verify_param_t * pVerifyPara`, `strcms_t * hStr`, `mem_blk_t * data`)

Функция продолжения потоковой проверки отделенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

`error_status_t VCERT1API VCERT_CmsStrDetVerifyFinalMem` (`context_t hCtx`, `verify_param_t * pVerifyPara`, `strcms_t * hStr`, `mem_blk_t * ocms`, `verify_result_t * pVerifyResult`)

Функция финализации потоковой проверки отделенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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** (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **pVerifyResult** (out) структура результата проверки ЭП CMS-сообщения (освобождается функцией **VCERT_FreeVerifyResult()** - см. описание в п. 1.31.1).

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**).

error_status_t VCERT1API **VCERT_CmsStrDetVerifyFile** (context_t hCtx, verify_param_t * pVerifyPara, const char * data, const char * icms, const char * ocms, verify_result_t * pVerifyResult)

Функция потоковой проверки отделенных ЭП файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeVerifyResult()** - см. описание в п. 1.31.1).

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**).

1.22 Зашифрование CMS-сообщений

Формат зашифрованных CMS-сообщений - CMS Enveloped - является универсальным за счет реализации с использованием стандарта ASN.1. Идентификация сертификата отправителя и сертификата(ов) получателей осуществляется либо посредством пары Имя издателя/Серийный номер, либо посредством использования данных дополнения "Идентификатор ключа владельца". Формат CMS Enveloped позволяет добавлять в CMS-сообщение дополнительные незащищенные атрибуты.

Для зашифрования CMS-сообщения прикладное ПО должно выполнить следующие действия:

- определить список сертификатов получателей зашифрованного сообщения:
 - либо передавая шаблоны поиска сертификатов получателей непосредственно в функции зашифрования. При возникновении ошибки поиска по од-

ному из шаблонов код этой ошибки возвращается функциями зашифрования. При невозможности использования ни одного из найденных сертификатов в качестве сертификата получателя возвращается код ошибки **VCERT_E_CERT_NOT_FOUND**;

- либо с помощью вызовов функции поиска сертификатов **VCERT_FindCert()** - по одному вызову для каждого шаблона поиска сертификатов получателей. При таком подходе рекомендуется, во избежание повторения процедуры поиска в функциях зашифрования, передавать в функции зашифрования блоки памяти с найденными сертификатами получателей в DER-кодировке;

- вызвать требуемые функции для собственно зашифрования CMS-сообщения. Функции зашифрования автоматически добавляют рабочий сертификат в список сертификатов получателей.

Перед зашифрованием CMS-сообщения функции зашифрования предварительно производят:

- проверку действительности сертификата отправителя (рабочего) и сертификатов получателей на текущий момент времени;
- проверку действительности закрытых ключей отправителя и получателей на текущий момент времени;
- проверку наличия в сертификате отправителя (рабочем) и в сертификатах получателей разрешенной области использования открытого ключа **KEYUSAGE_KEY_ENCIIPHERMENT** или **KEYUSAGE_KEY_AGREEMENT** (т.е. проверку возможности расшифрования на соответствующих закрытых ключах);
- добавление в список сертификатов получателей зашифрованного сообщения сертификата отправителя, в случае его отсутствия в этом списке.

При зашифровании в блочном режиме зашифрованное CMS-сообщение имеет ASN.1 кодировку определенной длины (ASN.1 definite-length encoding). При зашифровании в потоковом режиме зашифрованное CMS-сообщение имеет ASN.1 кодировку неопределенной длины (ASN.1 indefinite-length encoding).

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.

1.22.1 Структуры зашифрования CMS-сообщений

Структура **encrypt_param_t**

Структура параметров зашифрования CMS-сообщений:

- **flag_t flag**

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

- **FLAG_CMS_ENCRYPT_USEREMOTESEARCH** - аналогичен флагу **FLAG_FIND_USEREMOTESEARCH** (см. описание в п. 1.19.1);
- **FLAG_CMS_ENCRYPT_DONOTCHECKTIMES** - аналогичен флагу **FLAG_FIND_DONOTCHECKTIMES** (см. описание в п. 1.19.1);

- **FLAG_CMS_ENCRYPT_DONOTVERIFYCHAIN** - аналогичен флагу **FLAG_FIND_DONOTVERIFYCHAIN** (см. описание в п. 1.19.1);
- **FLAG_CMS_ENCRYPT_ADDREMOTETOLOCAL** - аналогичен флагу **FLAG_FIND_ADDREMOTETOLOCAL** (см. описание в п. 1.19.1);
- **FLAG_CMS_ENCRYPT_DONOTCHECKKEYTIME** - аналогичен флагу **FLAG_FIND_DONOTCHECKKEYTIME** (см. описание в п. 1.19.1);
- **FLAG_CMS_ENCRYPT_SUBJECTISPARTIAL** - аналогичен флагу **FLAG_FIND_SUBJECTISPARTIAL** (см. описание в п. 1.19.1);
- **FLAG_CMS_ENCRYPT_IGNORESTORECACHE** - аналогичен флагу **FLAG_FIND_IGNORESTORECACHE** (см. описание в п. 1.19.1);
- **FLAG_CMS_ENCRYPT_IGNORESTORELOCAL** - аналогичен флагу **FLAG_FIND_IGNORESTORELOCAL** (см. описание в п. 1.19.1);
- **FLAG_CMS_ENCRYPT_SUBJECTATTRIBUTE** - аналогичен флагу **FLAG_FIND_SUBJECTATTRIBUTE** (см. описание в п. 1.19.1);
- **FLAG_CMS_ENCRYPT_SUBJKEYID** - использовать для идентификации сертификатов получателей данные дополнений "Идентификатор ключа владельца". Если данный флаг не установлен, то для идентификации используются пары Имя издателя/Серийный номер;
- **FLAG_CMS_ENCRYPT_GOST_R_34_12_15MG** - использовать для зашифрования CMS-сообщений алгоритм ГОСТ Р 34.12-2015 (блочный шифр «Магма»). Данный флаг не может использоваться одновременно с флагом **FLAG_CMS_ENCRYPT_GOST_R_34_12_15GH**;
- **FLAG_CMS_ENCRYPT_GOST_R_34_12_15GH** - использовать для зашифрования CMS-сообщений алгоритм ГОСТ Р 34.12-2015 (блочный шифр «Кузнечик»). Если не установлен ни один из флагов **FLAG_CMS_ENCRYPT_GOST_R_34_12_15MG** и **FLAG_CMS_ENCRYPT_GOST_R_34_12_15GH**, то для зашифрования используется алгоритм ГОСТ 28147-89;
- **FLAG_CMS_ENCRYPT_ADDOMACATTRIBUTE** - вычислять и добавлять в зашифрованные CMS-сообщения имитовставку; если флаг не установлен, то имитовставка не добавляется. Должен быть установлен один из флагов **FLAG_CMS_ENCRYPT_GOST_R_34_12_15MG** или **FLAG_CMS_ENCRYPT_GOST_R_34_12_15GH**;
- **FLAG_CMS_ENCRYPT_RECIPKEYAGREEMENT** - использовать в CMS-сообщениях структуры получателей типа KeyAgreement; если флаг не установлен, то используются структуры получателей типа KeyTransport. Должен быть установлен один из флагов **FLAG_CMS_ENCRYPT_GOST_R_34_12_15MG** или **FLAG_CMS_ENCRYPT_GOST_R_34_12_15GH**.

– certid_t * **mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

– uint32_t **receiver_num**

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

– certificate_t * **receivers**

Поле с массивом шаблонов сертификатов получателей.

1.22.2 Функции блочного зашифрования CMS-сообщений

error_status_t VCERT1API **VCERT_CmsBlkEncryptMem** (context_t hCtx, encrypt_param_t * pEncryptPara, mem_blk_t * data, mem_blk_t * ocms)

Функция блочного зашифрования блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsBlkEncryptFile** (context_t hCtx, encrypt_param_t * pEncryptPara, const char * data, const char * ocms)

Функция блочного зашифрования файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.22.3 Функции потокового зашифрования CMS-сообщений

error_status_t VCERT1API **VCERT_CmsStrEncryptInitMem** (context_t hCtx, encrypt_param_t * pEncryptPara, mem_blk_t * data, strcms_t * hStr)

Функция инициализации потокового зашифрования блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_CmsStrEncryptUpdateMem** (context_t hCtx, encrypt_param_t * pEncryptPara, strcms_t * hStr, mem_blk_t * data, mem_blk_t * ocms)

Функция продолжения потокового зашифрования блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrEncryptFinalMem** (context_t hCtx, encrypt_param_t * pEncryptPara, strcms_t * hStr, mem_blk_t * ocms)

Функция финализации потокового зашифрования блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrEncryptFile** (context_t hCtx, encrypt_param_t * pEncryptPara, const char * data, const char * ocms)

Функция потокового зашифрования файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.23 Расшифрование CMS-сообщений

Функции расшифрования CMS-сообщений автоматически определяют:

- алгоритм, с помощью которого следует выполнять расшифрование - ГОСТ 28147-89, ГОСТ Р 34.12-2015 (блочный шифр «Магма»), ГОСТ Р 34.12-2015 (блочный шифр «Кузнечик»);

- тип структур получателей зашифрованного сообщения, содержащих зашифрованный сеансовый ключ - KeyTransport или KeyAgreement;

- наличие идентификатора рабочего сертификата в списке получателей зашифрованного сообщения - при его отсутствии возвращается код ошибки **VCERT_E_CMS_NOT_RECIPIENT**;

- наличие или отсутствие имитовставки в зашифрованном сообщении - при наличии имитовставки ее значение сравнивается с вычисленным и в случае несовпадения возвращается код ошибки **VCERT_E_CMS_OMAC_MISMATCH**.

Перед расшифрованием CMS-сообщения функции расшифрования предварительно производят:

- проверку действительности рабочего сертификата на текущий момент времени;

- проверку действительности закрытого ключа на текущий момент времени;

- проверку наличия в рабочем сертификате разрешенной области использования открытого ключа **KEYUSAGE_KEY_ENCRYPTION** или **KEYUSAGE_KEY_AGREEMENT** (т.е. проверку возможности расшифрования на закрытом ключе).

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.

1.23.1 Структуры расшифрования CMS-сообщений

Структура **decrypt_param_t**

Структура параметров расшифрования CMS-сообщений:

- **flag_t flag**

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

- **certid_t * mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

- **certinfo_t 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.

1.23.2 Функции блочного расшифрования CMS-сообщений

error_status_t VCERT1API VCERT_CmsBlkDecryptMem (**context_t hCtx**, **decrypt_param_t * pDecryptPara**, **mem_blk_t * icms**, **mem_blk_t * data**, **decrypt_result_t * pDecryptResult**)

Функция блочного расшифрования блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **pDecryptResult** (out) структура результата расшифрования CMS-сообщения (освобождается функцией **VCERT_FreeDecryptResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsBlkDecryptFile** (context_t hCtx, decrypt_param_t * pDecryptPara, const char * icms, const char * data, decrypt_result_t * pDecryptResult)

Функция блочного расшифрования файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeDecryptResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

1.23.3 Функции потокового расшифрования CMS-сообщений

error_status_t VCERT1API **VCERT_CmsStrDecryptInitMem** (context_t hCtx, decrypt_param_t * pDecryptPara, mem_blk_t * icms, strcms_t * hStr)

Функция инициализации потокового расшифрования блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_CmsStrDecryptUpdateMem** (context_t hCtx, decrypt_param_t * pDecryptPara, strcms_t * hStr, mem_blk_t * icms, mem_blk_t * data)

Функция продолжения потокового расшифрования блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrDecryptFinalMem** (context_t hCtx, decrypt_param_t * pDecryptPara, strcms_t * hStr, mem_blk_t * data, decrypt_result_t * pDecryptResult)

Функция финализации потокового расшифрования блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **pDecryptResult** (out) структура результата расшифрования CMS-сообщения (освобождается функцией **VCERT_FreeDecryptResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrDecryptFile** (context_t hCtx, decrypt_param_t * pDecryptPara, const char * icms, const char * data, decrypt_result_t * pDecryptResult)

Функция потокового расшифрования файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeDecryptResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

1.24 Преобразование совмещенных и отделенных ЭП

1.24.1 Функции блочного преобразования отделенных ЭП в совмещенные

`error_status_t VCERT1API VCERT_CmsBlkSignAttachMem (context_t hCtx, mem_blk_t * data, mem_blk_t * icms, mem_blk_t * ocms, flag_t flag)`

Функция блочного преобразования отделенных ЭП блока памяти в совмещенные

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

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

`error_status_t VCERT1API VCERT_CmsBlkSignAttachFile (context_t hCtx, const char * data, const char * icms, const char * ocms, flag_t flag)`

Функция блочного преобразования отделенных ЭП файла в совмещенные

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.24.2 Функции блочного преобразования совмещенных ЭП в отделенные

`error_status_t VCERT1API VCERT_CmsBlkSignDetachMem (context_t hCtx, mem_blk_t * icms, mem_blk_t * data, mem_blk_t * ocms, flag_t flag)`

Функция блочного преобразования совмещенных ЭП блока памяти в отделенные

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **icms** (in) блок памяти с подписанным CMS-сообщением с совмещенными ЭП (должно выполняться условие **icms->len > 0**);
- **data** (out) блок памяти с подписанными данными (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **ocms** (out) блок памяти с подписанным CMS-сообщением с отделенными ЭП (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsBlkSignDetachFile** (context_t hCtx, const char * icms, const char * data, const char * ocms, flag_t flag)

Функция блочного преобразования совмещенных ЭП файла в отделенные

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.24.3 Функции потокового преобразования совмещенных ЭП в отделенные

error_status_t VCERT1API **VCERT_CmsStrSignDetachInitMem** (context_t hCtx, mem_blk_t * icms, strcms_t * hStr, flag_t flag)

Функция инициализации потокового преобраз. совмещенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_CmsStrSignDetachUpdateMem** (context_t hCtx, strcms_t * hStr, mem_blk_t * icms, mem_blk_t * data)

Функция продолжения потокового преобраз. совмещенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrSignDetachFinalMem** (context_t hCtx, strcms_t * hStr, mem_blk_t * data, mem_blk_t * ocms)

Функция финализации потокового преобраз. совмещенных ЭП блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **hStr** (in) контекст потоковой операции;
- **data** (out) блок памяти с окончанием подписанных данных (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1);
- **ocms** (out) блок памяти с подписанным CMS-сообщением с отделенными ЭП (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrSignDetachFile** (context_t hCtx, const char * icms, const char * data, const char * ocms, flag_t flag)

Функция потокового преобразования совмещенных ЭП файла в отделенные

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

1.25 Получение информации о CMS-сообщениях

1.25.1 Структуры получения информации о CMS-сообщениях

Структура cms_certid_t

Структура идентификатора сертификата CMS-сообщения:

- uint32_t type

Поле с типом структуры идентификатора сертификатов CMS-сообщения:

- **FLAG_CMS_CERTID_TYPE_ISSN** - структура идентификатора серти-

фиката содержит пару Имя издателя/Серийный номер сертификата;

- **FLAG_CMS_CERTID_TYPE_SKID** - структура идентификатора сертификата содержит данные дополнения "Идентификатор ключа владельца".

- string_t **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_t **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_t **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_t **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_siginf_t

Структура информации о подписанте CMS-сообщения:

- uint32_t **version**

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

- string_t **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_t **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.

- date_t **signingTime**

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

- uint32_t **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**).*

– uint32_t **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:

– uint32_t **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.

– uint32_t **index**

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

– uint32_t **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_t **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_t **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:

– uint32_t **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).

– uint32_t **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**).

– uint32_t **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**).*

– uint32_t **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_t **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.

– uint32_t **unprotAttr_num**

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

– string_t * **unprotAttrs**

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

1.25.2 Функции блочного получения информации о CMS-сообщениях

error_status_t VCERT1API **VCERT_CmsBlkMsgInfMem** (context_t hCtx, mem_blk_t * cms, cms_msginf_t * pMsgInf, flag_t flag)

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

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **cms** (in) блок памяти с CMS-сообщением (должно выполняться условие **cms->len > 0**);
- **pMsgInf** (out) структура с информацией о CMS-сообщении (освобождается функцией **VCERT_CmsFreeMsgInf()** - см. описание в п. 1.31.1);
- **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsBlkMsgInfFile** (context_t hCtx, const char * cms, cms_msginf_t * pMsgInf, flag_t flag)

Функция блочного получения информации о CMS-сообщениях в виде файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **cms** (in) файл (не нулевой длины) с CMS-сообщением;

- **pMsgInf** (out) структура с информацией о CMS-сообщении (освобождается функцией **VCERT_CmsFreeMsgInf()** - см. описание в п. 1.31.1);
- **flag** (in) маска флагов (зарезервировано, должно быть равно 0).

Возвращаемые значения:

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

1.25.3 Функции потокового получения информации о CMS-сообщениях

error_status_t VCERT1API **VCERT_CmsStrMsgInfInitMem** (context_t hCtx, mem_blk_t * cms, strcms_t * hStr, flag_t flag)

Функция инициализации потокового получения информации о блоке памяти CMS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_CmsStrMsgInfUpdateMem** (context_t hCtx, strcms_t * hStr, mem_blk_t * cms)

Функция продолжения потокового получения информации о блоке памяти CMS

Аргументы:

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

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrMsgInfFinalMem** (context_t hCtx, strcms_t * hStr, cms_msginf_t * pMsgInf)

Функция финализации потокового получения информации о блоке памяти CMS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **hStr** (in) контекст потоковой операции;
- **pMsgInf** (out) структура с информацией о CMS-сообщении (освобождается функцией **VCERT_CmsFreeMsgInf()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_CmsStrMsgInfFile** (context_t hCtx, const char * cms, cms_msginf_t * pMsgInf, flag_t flag)

Функция потокового получения информации о CMS-сообщениях в виде файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **cms** (in) файл (не нулевой длины) с CMS-сообщением;
- **pMsgInf** (out) структура с информацией о CMS-сообщении (освобождается функцией **VCERT_CmsFreeMsgInf()** - см. описание в п. 1.31.1);
- **flag** (in) маска флагов (зарезервировано, должно быть равно **0**).

Возвращаемые значения:

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

1.26 Простановка и проверка штампов времени

Библиотека обеспечивает возможность простановки и проверки штампов времени ЭП в соответствии в RFC 3161 по протоколу Time Stamp Protocol (TSP). Штамп времени представляет собой подписанное CMS-сообщение, сформированное для конкретной ЭП электронного документа, которое обычно хранится в неаутентифицированном атрибуте **RFC 3161 timeStampToken** с объектным идентификатором 1.2.840.113549.1.9.16.2.14 указанной ЭП.

В простановке штампа времени участвуют две стороны - клиент и сервер штампов времени (Time Stamp Authority, TSA). После вычисления ЭП электронного документа клиент формирует запрос на простановку штампа времени для данной ЭП. В этот запрос клиент помещает хэш-значение ЭП (называемое также отпечатком ЭП), а также случайное число **nonce**, и отправляет его по протоколу **HTTP** для подписания на сервер штампов времени. Сервер на основании полученного запроса формирует подписанное CMS-сообщение с совмещенной ЭП, содержащее штамп времени, и отправляет его обратно клиенту. При наличии соответствующего признака в запросе клиента сервер включает в подписанное CMS-сообщение свой сертификат.

Требования к протоколу обмена клиента и сервера штампов времени:

- клиент должен формировать запрос на простановку штампа времени с отпечатком ЭП, вычисленным с использованием либо алгоритма хеширования ГОСТ Р 34.11-2012, 256 бит (объектный идентификатор алгоритма 1.2.643.7.1.1.2.2), либо ГОСТ Р 34.11-2012, 512 бит (объектный идентификатор алгоритма 1.2.643.7.1.1.2.3). Сервер должен отказать в обработке запроса на простановку штампа времени с отпечатком ЭП, вычисленным с использованием любого другого алгоритма хеширования;

- клиент должен включать в запрос на простановку штампа времени случайное число **nonce** (по умолчанию длиной **16** байт), позволяющее уникально идентифицировать данный запрос. Сервер должен отказать в обработке запроса на простановку штампа времени в случае отсутствия в нем указанного случайного числа;

- при вычислении ЭП штампа времени сервер должен скопировать в него отпечаток ЭП и случайное число **nonce** из запроса клиента. При получении штампа времени от сервера клиент должен сравнить отпечатки ЭП и случай-

ные числа в запросе и в штампе времени, и отклонить штамп времени при их несовпадении.

Требования к сертификату сервера штампов времени:

– дополнение "Разрешенная область использования открытого ключа" сертификата должно быть помечено как критичное, и в нем должны присутствовать исключительно использования **KEYUSAGE_DIGITAL_SIGNATURE** и **KEYUSAGE_NON_REPUDIATION** - т.е. открытый ключ разрешено использовать только для ЭП, но не для шифрования;

– дополнение "Расширенная область использования открытого ключа" сертификата должно быть помечено как критичное, и в нем должна присутствовать только одна расширенная область **RFC 3161 id-kp-timeStamping** с объектным идентификатором 1.3.6.1.5.5.7.3.8.

Проверка штампа времени должна проводиться после успешной проверки ЭП электронного документа, для которой он был проставлен. Процедура проверки штампа времени аналогична процедуре проверки ЭП подписанного CMS-сообщения, после которой выполняется проверка соответствия сертификата сервера штампов времени требованиям, описанным выше.

При простановке штампа времени в блочном режиме подписанное CMS-сообщение, включающее штамп времени, имеет ASN.1 кодировку определенной длины (ASN.1 definite-length encoding). При простановке штампа времени в потоковом режиме подписанное CMS-сообщение, включающее штамп времени, имеет ASN.1 кодировку неопределенной длины (ASN.1 indefinite-length encoding).

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.

1.26.1 Структуры простановки и проверки штампов времени

Структура `tsp_request_param_t`

Структура параметров простановки штампа времени CMS-сообщений:

– `flag_t` **flag**

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

• **FLAG_TSP_REQUEST_INCLUDENONCE** - добавлять в запрос на получение штампа времени случайное число **nonce** длиной 16 байт. Прикладное ПО должно использовать данный флаг при простановке штампов времени CMS-сообщений;

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

– `certid_t` * **mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

– `uint32_t` **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:

– flag_t **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**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

Структура tsp_verify_param_t

Структура параметров проверки штампа времени CMS-сообщений:

– flag_t **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**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

– uint32_t **index**

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

– certinfo_t **info**

Поле с требуемой возвращаемой информацией о сертификате сервера штампов времени.

Структура tsp_verify_result_t

Структура результата проверки штампа времени для заданной ЭП:

– date_t **time**

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

– certificate_t * **cert**

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

1.26.2 Функции блочной простановки и проверки штампов времени

error_status_t VCERT1API VCERT_TspBlkRequestFromCmsMem (context_t hCtx, tsp_request_param_t * pRequestPara, mem_blk_t * icms, mem_blk_t * oreq)

Функция блочного создания запроса на получение штампа времени блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TspBlkRequestFromCmsFile** (context_t hCtx, tsp_request_param_t * pRequestPara, const char * icms, mem_blk_t * oreq)

Функция блочного создания запроса на получение штампа времени файла

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TspSignResponse** (context_t hCtx, tsp_response_param_t * pResponsePara, mem_blk_t * ireq, mem_blk_t * ores)

Функция блочного вычисления ЭП и формирования штампов времени

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TspVerifyResponse** (context_t hCtx, tsp_verify_param_t * pVerifyPara, mem_blk_t * ires, tsp_verify_result_t * pVerifyResult)

Функция блочной проверки ЭП сформированного штампа времени

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_TspFreeVerifyResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TspBlkUrlStampCmsMem** (context_t hCtx, tsp_request_param_t * pRequestPara, const char * url, mem_blk_t * icms, mem_blk_t * ocms)

Функция блочной простановки штампа времени блока памяти CMS на сервере

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TspBlkUrlStampCmsFile** (context_t hCtx, tsp_request_param_t * pRequestPara, const char * url, const char * icms, const char * ocms)

Функция блочной простановки штампа времени файла CMS на сервере

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_TspBlkVerifyCmsMem** (context_t hCtx, tsp_

verify_param_t * pVerifyPara, mem_blk_t * icms, tsp_verify_result_t * pVerifyResult)

Функция блочной проверки штампа времени блока памяти CMS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_TspFreeVerifyResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TspBlkVerifyCmsFile** (context_t hCtx, tsp_verify_param_t * pVerifyPara, const char * icms, tsp_verify_result_t * pVerifyResult)

Функция блочной проверки штампа времени файла CMS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_TspFreeVerifyResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

1.26.3 Функции потоковой простановки и проверки штампов времени

error_status_t VCERT1API **VCERT_TspStrUrlStampCmsInitMem** (context_t hCtx, tsp_request_param_t * pRequestPara, const char * url, mem_blk_t * icms, strcms_t * hStr)

Функция инициализации потоковой простановки штампа времени CMS на сервере

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_TspStrUrlStampCmsUpdateMem** (context_t hCtx, tsp_request_param_t * pRequestPara, const char * url, strcms_t * hStr, mem_blk_t * icms, mem_blk_t * ocms)

Функция продолжения потоковой простановки штампа времени CMS на сервере

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TspStrUrlStampCmsFinalMem** (context_t hCtx, tsp_request_param_t * pRequestPara, const char * url, strcms_t * hStr, mem_blk_t * ocms)

Функция финализации потоковой простановки штампа времени CMS на сервере

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TspStrUrlStampCmsFile** (context_t hCtx,

tsp_request_param_t * pRequestPara, const char * url, const char * icms, const char * ocms)

Функция потоковой простановки штампа времени файла CMS на сервере

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_TspStrVerifyCmsInitMem** (context_t hCtx, tsp_verify_param_t * pVerifyPara, mem_blk_t * icms, strcms_t * hStr)

Функция инициализации потоковой проверки штампа времени блока памяти CMS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_TspStrVerifyCmsUpdateMem** (context_t hCtx, tsp_verify_param_t * pVerifyPara, strcms_t * hStr, mem_blk_t * icms)

Функция продолжения потоковой проверки штампа времени блока памяти CMS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

`error_status_t VCERT1API VCERT_TspStrVerifyCmsFinalMem` (`context_t hCtx`, `tsp_verify_param_t * pVerifyPara`, `strcms_t * hStr`, `tsp_verify_result_t * pVerifyResult`)
Функция финализации потоковой проверки штампа времени блока памяти CMS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_TspFreeVerifyResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

`error_status_t VCERT1API VCERT_TspStrVerifyCmsFile` (`context_t hCtx`, `tsp_verify_param_t * pVerifyPara`, `const char * icms`, `tsp_verify_result_t * pVerifyResult`)
Функция потоковой проверки штампа времени файла CMS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_TspFreeVerifyResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

1.27 Получение online-статуса сертификата

Библиотека обеспечивает возможность получения online-статуса сертификата в соответствии в RFC 6960 по протоколу Online Certificate Status Protocol (OCSP). Online-статус сертификата представляет собой подписанное сообщение - OCSP ответ (Response), сформированное для конкретного сертификата на конкретный момент времени.

В получении online-статуса сертификата участвуют две стороны - клиент и сервер OCSP ответчика (Responder). Клиент формирует запрос на получение online-статуса сертификата. В этот запрос клиент помещает идентификатор сертификата, online-статус которого требуется получить, а также случайную последовательность **nonce** - дополнение **RFC 6960 id-pkix-ocsp-nonce** с объектным идентификатором 1.3.6.1.5.5.7.48.1.2, и отправляет его по протоколу **HTTP** для обработки на сервер OCSP ответчика. Сервер на основании полученного запроса формирует подписанное сообщение, содержащее online-статус для указанного

сертификата, и отправляет его назад клиенту.

Требования к протоколу обмена клиента и сервера OCSP ответчика:

- клиент должен формировать запрос на получение online-статуса сертификата с идентификатором сертификата, вычисленным с использованием либо алгоритма хэширования ГОСТ Р 34.11-2012, 256 бит (объектный идентификатор алгоритма 1.2.643.7.1.1.2.2), либо ГОСТ Р 34.11-2012, 512 бит (объектный идентификатор алгоритма 1.2.643.7.1.1.2.3). Сервер должен отказать в обработке запроса на получение online-статуса сертификата, в котором идентификатор сертификата вычислен с использованием любого другого алгоритма хэширования;
- клиент должен включать в запрос на получение online-статуса сертификата случайную последовательность **nonce** (по умолчанию длиной **16** байт), позволяющую уникально идентифицировать данный запрос. Сервер должен отказать в обработке запроса на получение online-статуса сертификата в случае отсутствия в нем указанной случайной последовательности;
- при вычислении ЭП online-статуса сертификата сервер должен скопировать в него идентификатор сертификата и случайную последовательность **nonce** из запроса клиента. При получении online-статуса сертификата от сервера клиент должен сравнить идентификаторы сертификата и случайные последовательности в запросе и в online-статусе сертификата, и отклонить online-статус сертификата при их несовпадении.

Требования к сертификату сервера OCSP ответчика:

- дополнение "Разрешенная область использования открытого ключа" сертификата должно быть помечено как критичное, и в нем должны присутствовать использования **KEYUSAGE_DIGITAL_SIGNATURE** и **KEYUSAGE_NON_REPUDIATION** - т.е. открытый ключ разрешено использовать для ЭП;
- в дополнении "Расширенная область использования открытого ключа" должна присутствовать расширенная область **RFC 6960 id-kp-OCSPSigning** с объектным идентификатором 1.3.6.1.5.5.7.3.9.

1.27.1 Структуры получения online-статуса сертификата

Структура **ocsp_request_param_t**

Структура параметров получения online-статуса сертификата:

- flag_t **flag**

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

- certid_t * **mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

Структура **ocsp_response_param_t**

Структура параметров вычисления ЭП online-статуса сертификата:

- flag_t **flag**

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

- certid_t * **mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

Структура `ocsp_verify_param_t`

Структура параметров проверки ЭП online-статуса сертификата:

– `flag_t` **flag**

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

– `certid_t` * **mycert**

Указатель на структуру идентификатора сертификата для определения контекста выполнения (NULL при использовании локальной библиотеки).

– `certinfo_t` **info**

Поле с требуемой возвращаемой информацией о сертификате сервера OCSP ответчика.

Структура `ocsp_verify_result_t`

Структура результата проверки ЭП online-статуса сертификата:

– `int32_t` **status**

Поле с online-статусом сертификата:

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

– `int32_t` **reason**

Для аннулированного сертификата, поле с причиной аннулирования сертификата (см. описание в подразделе 1.8).

– `date_t` **revtime**

Для аннулированного сертификата, поле со временем аннулирования сертификата.

– `date_t` **thisupd**

Поле со временем начала действия данного online-статуса сертификата.

– `date_t` **nextupd**

Поле со временем окончания действия данного online-статуса сертификата.

– `certificate_t` * **cert**

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

1.27.2 Функции получения online-статуса сертификата

`error_status_t` VCERT1API **VCERT_OcspCreateRequest** (`context_t` hCtx, `ocsp_request_param_t` * pRequestPara, `mem_blk_t` * cert, `certificate_t` * rqst)

Функция создания запроса на получение online-статуса сертификата

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_OcspSignResponse** (context_t hCtx, ocsp_response_param_t * pResponsePara, mem_blk_t * rqst, mem_blk_t * resp)

Функция обработки запроса на получение online-статуса сертификата

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_OcspUrlObtainResponse** (context_t hCtx, ocsp_request_param_t * pRequestPara, const char * url, mem_blk_t * cert, mem_blk_t * resp)

Функция получения online-статуса сертификата на заданном OSCP сервере

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_OcspVerifyResponse** (context_t hCtx, ocsp_verify_param_t * pVerifyPara, mem_blk_t * resp, ocsp_verify_result_t * pVerifyResult)

Функция проверки ЭП online-статуса сертификата

Аргументы:

– **hCtx** (in) контекст библиотеки;

– **pVerifyPara** (in) структура параметров проверки ЭП online-статуса серти-

фикаата;

- **resp** (in) блок памяти с подписанным online-статусом сертификата (должно выполняться условие **resp->len > 0**);
- **pVerifyResult** (out) структура результата проверки ЭП online-статуса сертификата (освобождается функцией **VCERT_OcspFreeVerifyResult()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

1.28 Протокол безопасности транспортного уровня TLS 1.2

Библиотека обеспечивает возможность двум сторонам - клиенту и серверу - безопасно передавать друг другу данные посредством установления защищенного канала по протоколу Transport Layer Security (TLS) версия 1.2. Реализация протокола TLS библиотеки не предъявляет требований к способу передачи данных между клиентом и сервером - может быть использован обмен данными по сети, обмен блоками памяти, обмен файлами, и т.д.

При установлении защищенного канала клиент и сервер производят обмен служебными сообщениями (выполняют переговоры), причем клиент является инициатором начала такого обмена. Клиент и сервер обмениваются несколькими служебными сообщениями, формируемыми при вызове функции **VCERT_TlsSessionHandshake()**, причем количество пересылаемых друг другу сообщений может варьироваться. При обмене служебными сообщениями клиент и сервер должны выполнять проверку готовности защищенного канала посредством вызова функции **VCERT_TlsSessionComplete()**.

В процессе обмена служебными сообщениями стороны выполняют аутентификацию - либо клиент аутентифицирует сервер (односторонняя аутентификация), либо обе стороны аутентифицируют друг друга (двусторонняя аутентификация) - и вырабатывают сеансовые ключи для защиты передаваемых данных. Данные, передаваемые между клиентом и сервером, защищаются шифрованием и вычислением имитовставки.

При односторонней аутентификации клиент строит и проверяет цепочку сертификата сервера, при двусторонней аутентификации клиент и сервер строят и проверяют цепочки сертификатов друг друга.

Требования к сертификату клиента протокола TLS:

- дополнение "Разрешенная область использования открытого ключа" сертификата должно быть помечено как критичное, и в нем должно присутствовать использование **KEYUSAGE_DIGITAL_SIGNATURE** - т.е. открытый ключ разрешено использовать для ЭП;
- в дополнении "Расширенная область использования открытого ключа" должна присутствовать расширенная область **RFC 5246 id-kp-clientAuth** с объектным идентификатором 1.3.6.1.5.5.7.3.2.

Требования к сертификату сервера протокола TLS:

- дополнение "Разрешенная область использования открытого ключа" сертификата должно быть помечено как критичное, и в нем должно присутствовать использование **KEYUSAGE_KEY_AGREEMENT** - т.е. открытый ключ разрешено

использовать для шифрования;

– в дополнении "Расширенная область использования открытого ключа" должна присутствовать расширенная область **RFC 5246 id-kp-serverAuth** с объектным идентификатором 1.3.6.1.5.5.7.3.1;

– в дополнении "Альтернативное имя владельца" должно присутствовать DNS-имя сервера (необходимо при верификации DNS-имени сервера клиентом).

1.28.1 Функции протокола безопасности транспортного уровня TLS 1.2

error_status_t VCERT1API VCERT_TlsSessionCreate (context_t hCtx, const char * server, tls_t * hTls, flag_t flag)

Функция создания контекста защищенного канала по протоколу TLS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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_CREAT_CLIENT_SESSION** - создавать контекст защищенного канала по протоколу TLS клиента. Если данный флаг не установлен, создается контекст сервера;

• **FLAG_TLS_ACCEPT_CLIENT_CERTIF** - принимать сертификат клиента для построения и проверки цепочки (может использоваться только при создании контекста сервера). При установке данного флага возможны как односторонняя, так и двусторонняя аутентификация;

• **FLAG_TLS_REQUIRE_CLIENT_CERTIF** - требовать сертификат клиента для построения и проверки цепочки (может использоваться только при создании контекста сервера). При установке данного флага проводится обязательная двусторонняя аутентификация;

• **FLAG_TLS_NO_RENEGOTIATION** - не использовать ранее установленный защищенный канал при выполнении переподключения (может использоваться только при создании контекста сервера);

• **FLAG_TLS_PROTOCOL_VERSION_1_2** - включить использование протокола TLS версия 1.2. По умолчанию, клиентская сторона предлагает использовать протокол TLS версия 1.0, а серверная сторона принимает протокол TLS версий 1.2 и 1.0.

Возвращаемые значения:

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

error_status_t VCERT1API VCERT_TlsSessionDestroy (context_t hCtx, tls_t hTls)

Функция освобождения контекста защищенного канала по протоколу TLS

Аргументы:

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

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TlsSessionHandshake** (context_t hCtx, tls_t hTls, mem_blk_t * itls, mem_blk_t * otls)

Функция выполнения переговоров при создании защищенного канала TLS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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** ≥ 0). При формировании первого сообщения клиентом длина должна быть установлена в 0;
- **otls** (out) блок памяти с данными протокола TLS, отправляемыми противоположной стороне (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TlsSessionComplete** (context_t hCtx, tls_t hTls)

Функция определения состояния защищенного канала по протоколу TLS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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.

error_status_t VCERT1API **VCERT_TlsSessionQuery** (context_t hCtx, tls_t hTls, uint32_t query, mem_blk_t * data)

Функция получения данных или управления защищенным каналом TLS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **hTls** (in) контекст защищенного канала по протоколу TLS;
- **query** (in) запрашиваемые данные или команда управления:

• **VCERT_TLS_QRY_PEER_CERTIF_BLOB** - запросить сертификат противоположной стороны в DER-кодировке (при его наличии возвращается в блок памяти **data**);

• **VCERT_TLS_QRY_PROTOCOL_VERSION** - получить версию протокола TLS в виде числа (возвращается в блок памяти **data**):

- **0x0301** - версия 1.0;
- **0x0302** - версия 1.1;
- **0x0303** - версия 1.2;

• **VCERT_TLS_QRY_CIPH_SUITE_VALUE** - получить идентификатор криптографического набора протокола TLS в виде числа (возвращается в блок памяти **data**):

- **0xFF85** - набор на базе ГОСТ 28147-89, ГОСТ Р 34.11-2012 и ГОСТ Р 34.10-2012;
- **0xFF89** - набор на базе ГОСТ Р 34.12-2015 (блочный шифр «Кузнечик»), ГОСТ Р 34.11-2012 и ГОСТ Р 34.10-2012;

• **VCERT_TLS_QRY_CIPH_SUITE_DESCR** - получить описание криптографического набора протокола TLS в виде строки (возвращается в блок памяти **data**):

- **TLS_GOST_R_3410_12_WITH_28147_CNT_IMIT** - набор на базе ГОСТ 28147-89, ГОСТ Р 34.11-2012 и ГОСТ Р 34.10-2012;
- **TLS_GOST_R_3410_12_WITH_GOST_R_3412_15GH_CTR_OMAC** - набор на базе ГОСТ Р 34.12-2015 (блочный шифр «Кузнечик»), ГОСТ Р 34.11-2012 и ГОСТ Р 34.10-2012;

• **VCERT_TLS_QRY_SESSION_SHUTDOWN** - отправить противоположной стороне оповещение о корректном закрытии защищенного канала **TLS CloseNotify Alert** (блок памяти **data** не заполняется);

– **data** (out) блок памяти с запрошенными данными или результатом команды управления (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TlsSessionWrite** (context_t hCtx, tls_t hTls, mem_blk_t * itls, mem_blk_t * otls)

Функция отправки данных другой стороне защищенного канала TLS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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** \geq 0);
- **otls** (out) блок памяти с данными протокола TLS, отправляемыми противоположной стороне (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

error_status_t VCERT1API **VCERT_TlsSessionRead** (context_t hCtx, tls_t hTls, mem_blk_t * itls, mem_blk_t * otls)

Функция получения данных от другой стороны защищенного канала TLS

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **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 (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

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.

1.29 Преобразование в формат и из формата Base64

Функции преобразования бинарных данных в формат и из формата Base64 можно использовать для представления защищенных (подписанных или зашифрованных) данных в текстовом виде для хранения в текстовых документах (например, в XML-документах).

Функция закодирования бинарных данных в формат Base64 не возвращает в закодированных данных символы, не входящие в алфавит Base64. Функция декодирования бинарных данных из формата Base64 завершается с ошибкой при наличии в закодированных данных символов, не входящих в алфавит Base64. Алфавит Base64 описан в RFC 4648 (см. статью на английском языке по ссылке <https://tools.ietf.org/rfc/rfc4648.txt>).

1.29.1 Функции преобразования в формат и из формата Base64

size_t VCERT1API **VCERT_EncodeMem** (unsigned char * t, const unsigned char * f, size_t 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 кодирования;
- **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.

size_t VCERT1API **VCERT_DecodeMem** (unsigned char * t, const unsigned char * f, size_t 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.

1.30 Выработка случайного числа заданной длины

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

1.30.1 Функции выработки случайного числа заданной длины

error_status_t VCERT1API **VCERT_GenRandom** (context_t hCtx, uint32_t size, mem_blk_t * rand)

Функция выработки случайного числа заданной длины

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **size** (in) длина случайного числа для выработки в байтах;
- **rand** (out) блок памяти для записи случайного числа (освобождается функцией **VCERT_FreeMem()** - см. описание в п. 1.31.1).

Возвращаемые значения:

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

1.31 Выделение и освобождение памяти

В результате своего выполнения функции библиотеки могут выделять блоки памяти или более сложные структуры для передачи результата прикладному ПО. Блоки памяти или структуры, выделенные в функциях библиотеки, должны

быть освобождены с помощью одной из функций библиотеки, предназначенных для освобождения памяти. Игнорирование данного требования, а также попытка освобождения блоков памяти или структур с помощью функций сторонних библиотек - например, системных - приведет к утечке памяти или аварийному завершению процесса прикладного ПО.

1.31.1 Функции выделения и освобождения памяти

`void * VCERT1API VCERT_AllocMem (context_t hCtx, size_t len)`

Функция выделения блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **len** (in) требуемый размер блока памяти в байтах.

Возвращаемые значения:

- **!= NULL** указатель на выделенный блок памяти в случае успеха.

`void * VCERT1API VCERT_ReallocMem (context_t hCtx, void * data, size_t len)`

Функция пере-выделения блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **data** (in) указатель на существующий блок памяти;
- **len** (in) новый требуемый размер блока памяти в байтах.

Возвращаемые значения:

- **!= NULL** указатель на пере-выделенный блок памяти в случае успеха.

`void VCERT1API VCERT_FreeMem (context_t hCtx, void * data)`

Функция освобождения блока памяти

Аргументы:

- **hCtx** (in) контекст библиотеки (при анализе ошибки, возникшей в функции инициализации, разрешено подавать значение контекста равным NULL);
- **data** (in) указатель на освобождаемый блок памяти.

`error_status_t VCERT1API VCERT_CopyCert (context_t hCtx, certificate_t * icer, certificate_t * ocer)`

Функция копирования содержимого структуры сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **icer** (in) указатель на исходную структуру сертификата;
- **ocer** (out) указатель на структуру сертификата назначения (дублируется только содержимое структуры, но не сама структура) (освобождается функцией **VCERT_FreeCert()**).

Возвращаемые значения:

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

`void VCERT1API VCERT_FreeCertId (context_t hCtx, certid_t * pCertId)`

Функция освобождения содержимого структуры уникального идентификатора сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pCertId** (in) указатель на освобождаемую структуру уникального идентификатора сертификата (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_FreeCert** (context_t hCtx, certificate_t * pCert)

Функция освобождения содержимого структуры сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pCert** (in) указатель на освобождаемую структуру сертификата (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_FreeCrl** (context_t hCtx, crl_t * pCrl)

Функция освобождения содержимого структуры САС

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pCrl** (in) указатель на освобождаемую структуру САС (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_FreeAltnameEx** (context_t hCtx, altname_ex_t * pAltname)

Функция освобождения содержимого структуры массива других имен альтернативного имени

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pAltname** (in) указатель на освобождаемую структуру массива других имен альтернативного имени (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_FreeVerifyResult** (context_t hCtx, verify_result_t * pVerifyResult)

Функция освобождения содержимого структуры результата проверки ЭП

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pVerifyResult** (in) указатель на освобождаемую структуру результата проверки ЭП (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_FreeDecryptResult** (context_t hCtx, decrypt_result_t * pDecryptResult)

Функция освобождения содержимого структуры результата расшифрования

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pDecryptResult** (in) указатель на освобождаемую структуру результата расшифрования (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_FreeFindResult** (context_t hCtx, find_result_t *

pFindResult)

Функция освобождения содержимого структуры результата поиска сертификатов

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pFindResult** (in) указатель на освобождаемую структуру результата поиска сертификатов (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_CmsFreeMsgInf** (context_t hCtx, cms_msginf_t * pMsgInf)

Функция освобождения содержимого структуры информации о CMS сообщении

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pMsgInf** (in) указатель на освобождаемую структуру информации о CMS-сообщении (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_TspFreeVerifyResult** (context_t hCtx, tsp_verify_result_t * pVerifyResult)

Функция освобождения содержимого структуры результата проверки штампа времени

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pVerifyResult** (in) указатель на освобождаемую структуру результата проверки штампа времени (освобождается только содержимое структуры, но не сама структура).

void VCERT1API **VCERT_OcspFreeVerifyResult** (context_t hCtx, ocsp_verify_result_t * pVerifyResult)

Функция освобождения содержимого структуры результата проверки статуса сертификата

Аргументы:

- **hCtx** (in) контекст библиотеки;
- **pVerifyResult** (in) указатель на освобождаемую структуру результата проверки статуса сертификата (освобождается только содержимое структуры, но не сама структура).

1.32 Описание конфигурационного файла pki1.conf

Конфигурационный файл **pki1.conf** предназначен для хранения информации о настроенных профилях пользователя. Данный файл может использоваться при вызове упрощенной функции инициализации контекста библиотеки **VCERT_Initialize()**.

Конфигурационный файл **pki1.conf** представляет собой текстовый файл в кодировке CP1251, содержащий строки вида **<Directive>: <Value>**, где **<Directive>** - идентификатор директивы (конфигурационного параметра), а

<Value> - значение директивы (конфигурационного параметра). Строки, начинающиеся с символа **#**, являются комментариями.

В файле **pki1.conf** могут использоваться следующие директивы:

- **default** задает имя профиля пользователя, используемого по умолчанию (директива опциональна);
- **local** задает имя профиля пользователя, к которому относятся последующие директивы (директива обязательна);
- **flag** задает маску (побитовое ИЛИ) флагов инициализации контекста библиотеки (см. описание в п. 1.9.1) (директива опциональна);
- **pse** задает путь (URI) к ПСП профиля пользователя (см. описание в п. 1.9.1) (директива обязательна);
- **localstore** задает путь (URI) к ЛСП профиля пользователя (см. описание в п. 1.9.1) (директива обязательна);
- **ldap** задает путь (URI) к ССС профиля пользователя (см. описание в п. 1.9.1) (директива опциональна);
- **pincode** задает ПИН-код для доступа к ключевому носителю типа смарт-карта (директива опциональна).

Ниже приведен пример конфигурационного файла **pki1.conf**:

```
default: По умолчанию
# Конфигурация профиля "По умолчанию"
local: По умолчанию
flag: 1
pse: pse://signed/C:\Профили\По умолчанию\local.pse
localstore: odbc://DEFAULT
ldap: ldap://ldap.site.ru:389/DC=site,DC=ru
pincode: 12345678
```

1.33 Описание XML-шаблона и XML-запроса

XML-шаблон и XML-запрос представляют собой текстовые XML-документы в кодировке CP1251, содержащие информацию, достаточную для формирования PKCS#10 запроса на получение сертификата пользователя. XML-запрос отличается от XML-шаблона только тем, что дополнительно содержит открытый ключ пользователя, предназначенный для включения в формируемый PKCS#10 запрос.

Ниже приведен пример XML-запроса без необходимой декларации XML-документа (без **XML Declaration**):

```
<pkiUser>
  <subject>INNLE=7700040698,OGRN=5087746616832,SNILS=98765432109,T=
    Главный специалист,SN=Родионов,GN=Алексей Николаевич,CN=Открытое
    акционерное общество Банк Москвы,0=0A0 "Банк Москвы",street=ул.
    Рождественка, д.8/15, стр.3,L=Москва,ST=77 г. Москва,C=RU</
    subject>
  <subjectAltName>
```

```

<emailAddress>rodionov@bm.ru</emailAddress>
<DNS>rodionov.bm.ru</DNS>
<URI>http://rodionov.bm.ru</URI>
<IP>11.22.33.44</IP>
<organizationName>ОАО "Банк Москвы"</organizationName>
<registredAddress>г. Москва, ул. Рождественка, д.8/15, стр.3</
  registredAddress>
<surname>Родионов Алексей Николаевич</surname>
<businessCategory>Главный специалист</businessCategory>
<telephoneNumber>+7(495)123-4567</telephoneNumber>
<description>Сертификат должностного лица</description>
<accountNumber>5647382910</accountNumber>
<BIC>5087746616832</BIC>
<physicalDelivery>Головной офис</physicalDelivery>
<exchangeAddress>/CN=RodionovAlexey/O=BankMoskvui/DC=pki/DC=bm/
  DC=ru</exchangeAddress>
<notesAddress>/CN=RodionovAlexey/O=BankMoskvui/DC=pki/DC=bm/DC=
  ru</notesAddress>
</subjectAltName>
<Policy>
  <OID>1.2.643.100.113.1</OID>
</Policy>
<Policy>
  <OID>1.2.643.100.113.2</OID>
</Policy>
<ExtKeyUsage>1.3.6.1.5.5.7.3.4</ExtKeyUsage>
<ExtKeyUsage>1.3.6.1.5.5.7.3.2</ExtKeyUsage>
<Extension>
  <OID>1.3.6.1.4.1.10244.4.5</OID>
  <Type>ASN1_IA5STRING</Type>
  <Value>rodionov@bm.ru</Value>
</Extension>
<Extension>
  <OID>1.2.643.100.111</OID>
  <Type>ASN1_UTF8STRING</Type>
  <Value>СКЗИ "Валидата CSP" версия 6.0</Value>
</Extension>
<Encipherment>yes</Encipherment>
<UsagePeriod>
  <KeyLifeTime>15</KeyLifeTime>
</UsagePeriod>
<KeyName>1234ABCDEF01</KeyName>
<PublicKey>06:20:00:00:49:2E:00:00:4D
  :41:47:31:00:02:00:00:30:12:06:07:2A:85:03:02:02:24:00:06:07:2A
  :85:03:02:02:1E:01:C5:C3:98:B4:D5:8F:A4:AF:83:F7:F1:D4:63:2E:D3
  :51:97:16:FC:8E:8C:53:FD:33:2E:1F:FA:2B:CD:25:B2:0E:23:65:D8
  :85:83:E0:0E:5B:47:1A:A6:5F:39:41:5E:D4:3E:6B:8F:4F:DD:79:C9:AE:
  D2:DB:CA:DB:0E:76:EE:B9</PublicKey>

```

</pkiUser>

Корневым элементом XML-документа является элемент **pkiUser**.

Строка с X.500-именем владельца сертификата задается в элементе **pkiUser -> subject**. Данный элемент является обязательным и должен присутствовать в XML-документе. Описание элементов (составных частей) X.500-имени владельца сертификата приведено ниже:

- **INN** - идентификационный номер налогоплательщика физического лица;
- **INNLE** - идентификационный номер налогоплательщика юридического лица;
- **OGRN** - основной государственный регистрационный номер;
- **OGRNIP** - основной государственный регистрационный номер для индивидуального предпринимателя;
- **SNILS** - страховой номер индивидуального лицевого счета;
- **T** - наименование должности лица;
- **SN** - фамилия физического лица *Примечание: фамилия уполномоченного представителя;*
- **GN** - приобретенное имя — имя и отчество физического лица *Примечание: имя и отчество уполномоченного представителя;*
- **CN** - общее имя;
- **OU** - наименование подразделения юридического лица;
- **O** - наименование юридического лица;
- **street** - адрес места нахождения соответствующего лица, включающий наименование улицы, номер дома, а также корпуса, строения, квартиры, помещения (при наличии);
- **L** - наименование соответствующего населенного пункта;
- **ST** - наименование соответствующего субъекта Российской Федерации;
- **C** - наименование страны — двухсимвольный код страны;
- **email** - адрес электронной почты RFC 822;
- **unstructuredName** - неструктурированное имя;
- **unstructuredAddress** - неструктурированный адрес;
- **DC** - доменный компонент RFC 2247.

Альтернативное имя владельца сертификата задается в элементе **pkiUser -> subject -> subjectAltName**. Альтернативное имя владельца сертификата может задаваться как полностью, так и частично, или может вообще отсутствовать в XML-документе. Описание вложенных элементов альтернативного имени владельца сертификата приведено ниже:

- **emailAddress** - строка с адресом электронной почты RFC 822;
- **DNS** - строка с именем системы имен доменов DNS;
- **URI** - с уникальным адресом ресурса URI;
- **IP** - адресом IP протокола;

- **organizationName** - с наименованием организации;
- **registredAddress** - с зарегистрированным адресом;
- **surname** - строка с Ф.И.О. владельца сертификата;
- **businessCategory** - строка с должностью владельца сертификата;
- **telephoneNumber** - строка с номером телефона владельца сертификата;
- **description** - строка с описанием в свободной форме;
- **accountNumber** - строка с номером расчетного счета;
- **BIC** - строка с БИК;
- **physicalDelivery** - с почтовым адресом;
- **exchangeAddress** - с адресом Microsoft Exchange;
- **notesAddress** - строка с адресом Lotus Notes;
- **UPN** - строка с именем участника-пользователя Microsoft.

Регламенты (политики) использования сертификата задаются посредством объектных идентификаторов (OID) в элементах **pkiUser -> Policy -> OID**. Таких элементов может быть несколько, или они вообще могут отсутствовать в XML-документе.

Расширенные области использования открытого ключа сертификата задаются посредством объектных идентификаторов (OID) в элементах **pkiUser -> ExtKeyUsage**. Таких элементов может быть несколько, или они вообще могут отсутствовать в XML-документе.

Дополнения (расширения) сертификата задаются в элементах **pkiUser -> Extension**. Таких элементов может быть несколько, или они вообще могут отсутствовать в XML-документе.

Объектные идентификаторы расширений задаются в элементах **pkiUser -> Extension -> OID**, типы расширений (под типом расширения подразумевается тип используемой для кодирования ASN.1 строки: **ASN1_IA5STRING** для IA5String, **ASN1_UTF8STRING** - для UTF8String) - в элементах **pkiUser -> Extension -> Type**, а значения расширений в виде строк - в элементах **pkiUser -> Extension -> Value**.

Если открытый ключ пользователя разрешено использовать для шифрования, то в XML-документе должен присутствовать элемент **pkiUser -> Encipherment**, значение которого должно быть установлено в **yes**. Если данный элемент отсутствует или имеет значение **no**, то использование открытого ключа пользователя для шифрования не разрешено.

Срок действия закрытого ключа пользователя, соответствующего выпускаемому сертификату, задается (в месяцах) в элементе **pkiUser -> UsagePeriod -> KeyLifetime**. Если данный элемент отсутствует, то срок действия закрытого ключа считается равным **15** месяцам.

Открытый ключ пользователя содержится в элементе (опциональном) **pkiUser -> PublicKey**. В этом элементе должна присутствовать шестнадцатеричная текстовая строка (**hex-string**) байтов открытого ключа в представлении Microsoft CryptoAPI **PUBLICKEYBLOB**. Если данный элемент присутствует в XML-документе, то в опциональном элементе **pkiUser -> KeyName** можно указать строку идентификатора закрытого ключа пользователя, соответствующего выпускаемому сертификату.

2 ОПИСАНИЕ ОШИБОЧНЫХ СИТУАЦИЙ

Ниже (Таблица 4) 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 4 – 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 хеширования
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

Имя и код ошибки	Описание и причина возникновения ошибки
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
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

Имя и код ошибки	Описание и причина возникновения ошибки
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)	CAC издателя не был найден в доступных справочниках. В доступных справочниках отсутствует CAC, необходимый для построения цепочки, при этом не разрешен или отсутствует доступ к точкам CDP
VCERT_E_CRL_EXPIRED (0xE070003C)	Срок действия CAC уже истек
VCERT_E_CRL_DAMAGED (0xE070003D)	CAC повреждён 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
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 CAC. Вероятно, при критичном обновлении одного из CAC, 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 истёк
VCERT_E_KEY_NOT_YET_VALID (0xE070004D)	Ключ ЭП/закрытый ключ шифрования еще недействителен
VCERT_E_CRL_NOT_YET_VALID (0xE070004E)	Срок действия CAC 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

Имя и код ошибки	Описание и причина возникновения ошибки
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
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

Имя и код ошибки	Описание и причина возникновения ошибки
VCERT_E_T00_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 ПСП 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
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

Имя и код ошибки	Описание и причина возникновения ошибки
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

Имя и код ошибки	Описание и причина возникновения ошибки
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
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

Имя и код ошибки	Описание и причина возникновения ошибки
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
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

Имя и код ошибки	Описание и причина возникновения ошибки
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_- SYTIME (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
КС	Криптографический сервер
ЛСП	Локальный справочник пользователя (Local Certificate Store)
ОС	Операционная система (Operating System)
ПК	Программный комплекс
ПО	Программное обеспечение
ППО	Прикладное ПО
ПСП	Персональный справочник пользователя (Personal Security Environment)
САС	Список аннулированных сертификатов (Certificate Revocation List)
СЗИ	Средство защиты информации
СКЗИ	Средство криптографической защиты информации
ССС	Сетевой справочник сертификатов (Network Certificate Store)
СУС	Система управления сертификатами (Public Key Infrastructure)
ФКН	Функциональный ключевой носитель
ЦР	Центр регистрации (Registration Authority)
ЦС	Центр сертификации (Certification Authority)
ЭД	Электронный документ
ЭП	Электронная подпись (Digital Signature)

ПЕРЕЧЕНЬ РИСУНКОВ

1	Верификация цепочки сертификации	43
---	--	----

ПЕРЕЧЕНЬ ТАБЛИЦ

1	Описание параметров конфигурации	12
2	Результаты проверки ЭП на аннулированном сертификате (без использования флага FLAG_CMS_VERIFY_USEREVOCATIONTIME)	60
3	Результаты проверки ЭП на аннулированном сертификате (с использованием флага FLAG_CMS_VERIFY_USEREVOCATIONTIME)	60
4	Описание ошибочных ситуаций	109

[illegible]