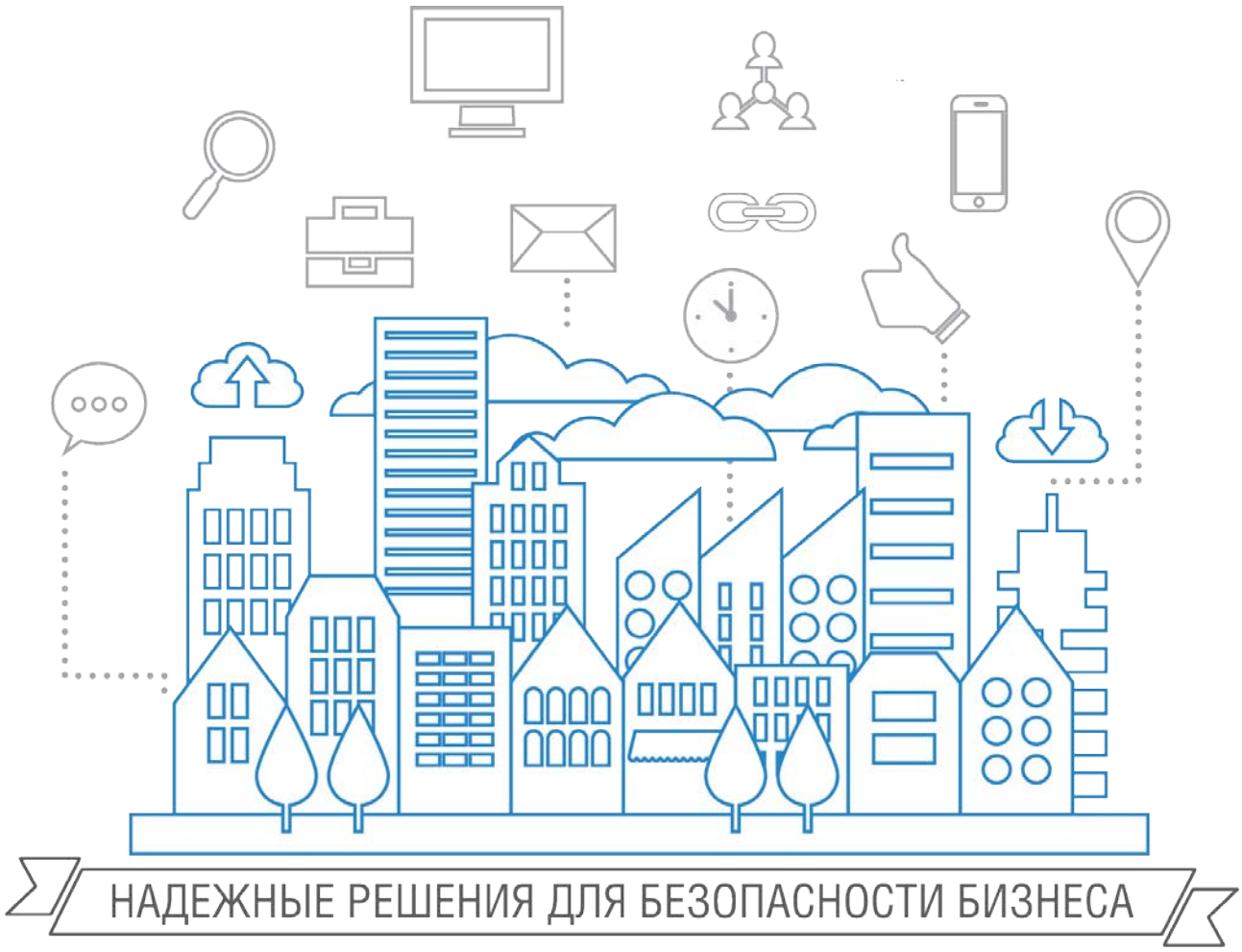
ГАЗИНФОРМСЕРВИС

198096, г. Санкт-Петербург, ул. Кронштадтская, д.10, лит. А, тел.: (812) 677-20-50, факс: (812) 677-20-51
Почтовый адрес: 198096, г. Санкт-Петербург, а/я 59, e-mail: resp@gaz-is.ru, www.gaz-is.ru
р/с 40702810800000001703 Ф-л Банка ГПБ (АО) в г. Санкт-Петербурге БИК 044030827,
к/с 30101810200000000827, ОКПО 72410666, ОГРН 1047833006099, ИНН/КПП 7838017968/783450001

Криптографическая платформа «Litoria Crypto Platform»

Руководство системного программиста



Санкт-Петербург, 2017

Аннотация

В документе приводится руководство системного программиста криптографической платформы «Litoria Crypto Platform».

В разделе «Общие сведения о криптоплатформе» описано назначение криптоплатформы, перечислены функции криптоплатформы и приведены краткие описания основных функций, указано программное обеспечение, которое необходимо для функционирования платформы «Litoria Crypto Platform», и приведены входные параметры для функций криптоплатформы и выходные параметры, которые формирует криптоплатформа.

В разделе «Структура криптоплатформы» приведены заголовки объявления, входные и выходные параметры, а также возвращаемые значения импортируемых функций платформы «Litoria Crypto Platform».

В разделе «Сообщения программисту» описан способ получения информации об ошибке и приведены коды ошибок, возвращаемые криптоплатформой.

В конце документа приведен список использованных сокращений.

Содержание

1 Общие сведения о криптоплатформе

Криптографическая платформа «Litoria Crypto Platform» предназначена для выполнения операций шифрования и создания усовершенствованной электронной подписи (УЭП) файловых объектов, отправки и получения запросов по сети для получения штампов времени (TSP) и OCSP-ответов, получение некоторой дополнительной информации о сертификатах, например, адреса OCSP служб (http://www.ietf.org/rfc/rfc2560). Криптоплатформа предоставляет все необходимые сервисы DVCS для обеспечения услуг ДТС – формирование запросов всех типов, формирование и анализ квитанции и т.д. в соответствии с документом RFC3029 (http://www.ietf.org/rfc/rfc3029).

Криптоплатформа обеспечивает возможность обращения к функциям криптопровайдеров, реализованных в соответствии с технологией Microsoft CSP.

Криптографическая платформа «Litoria Crypto Platform» выполняет следующие основные функции:

- создание УЭП;
- добавление УЭП;
- заверка УЭП;
- проверка УЭП;
- шифрование;
- расшифровывание;
- создание и отправка 4-х типов DVCS-запросов;
- формирование DVC-квитанции;
- анализ DVC-квитанции и вывод информации.

1.1 Описание функций криптоплатформы «Litoria Crypto Platform»

1.1.1 Создание УЭП

Электронная подпись (ЭП) – реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа ЭП и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе.

Сертификат ключа подписи (далее – сертификат) – документ на бумажном носителе или электронный документ с электронной подписью уполномоченного лица удостоверяющего центра, которые включают в себя открытый ключ электронной подписи и которые выдаются удостоверяющим центром участнику информационной системы для подтверждения подлинности электронной подписи и идентификации владельца сертификата ключа подписи.

УЭП – электронная подпись, усовершенствованная, в качестве меры борьбы с общепризнанными угрозами безопасности, добавлением (как необходимое требование) признаков ее (подписи) регламента и доказательств подлинности – таких, как штамп времени, данные об отзыве сертификата и др.

Для создания ЭП должен быть осуществлен выбор сертификата открытого ключа и параметры создания ЭП. К таким параметрам относятся:

- добавление сертификата подписывающего лица в подпись;
- добавление штампа времени;
- создание УЭП;
- выбор формата файла на выходе.

Общая схема выполнения процедуры создания УЭП и схема соответствующих импортируемых функций криптоплатформы, которые нужно вызвать для реализации того или иного действия, приведены на рисунке 1.1.

Рисунок 1.1. Схема операции создания УЭП

1.1.2 Добавление УЭП

Если в подписании документа участвует несколько лиц, и каждый должен поставить в нем свою подпись (например, в листе согласования), используется операция добавления УЭП.

В отличие от операции создания УЭП добавление подписи производится в уже подписанный ранее документ.

С помощью криптографической платформы «Litoria Crypto Platform» можно добавить подпись, созданную на криптографическом алгоритме, отличном от ГОСТ, например, RSA или AES. Основной областью применения такой подписи является система доверенной третьей стороны. Среди отличительных особенностей таких систем можно выделить необходимость юридически значимого документооборота между различными странами или регионами, в которых используются национальные криптографические стандарты.

Общая схема выполнения процедуры добавления УЭП и схема соответствующих импортируемых функций криптоплатформы, которые нужно вызвать для реализации того или иного действия, приведены на рисунке 1.2.

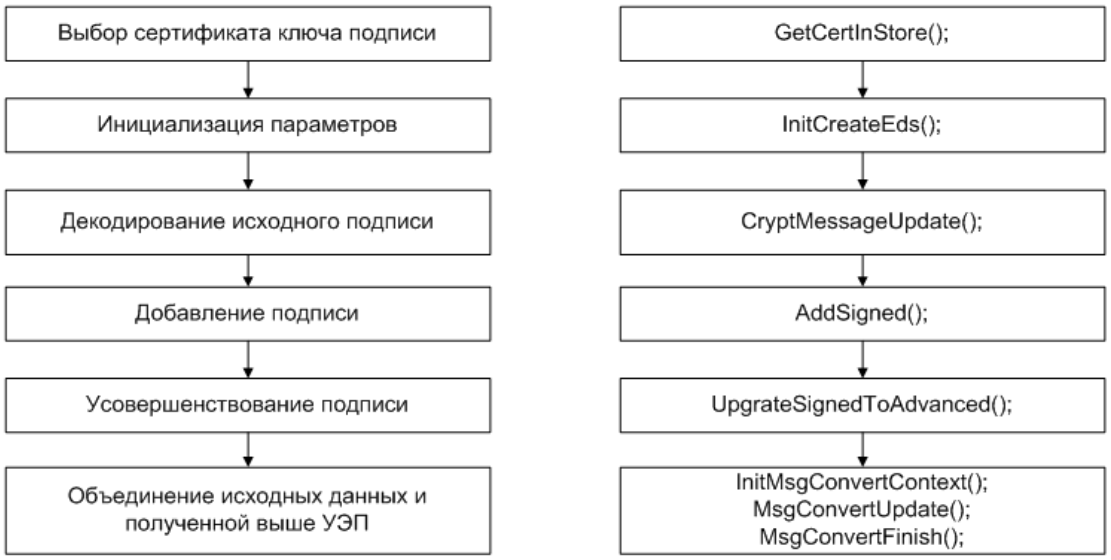


Рисунок 1.2. Схема операции добавления УЭП

1.1.3 Проверка УЭП

Проверка УЭП подразумевает подтверждение подлинности ЭП в электронном документе, то есть подтверждение:

- принадлежности ЭП в электронном документе владельцу сертификата ключа;
- отсутствия искажений в подписанном данной ЭП электронном документе;
- момента ЭП;
- действительности сертификата на момент создания ЭП.

Проверка УЭП файла с отделенной подписью – проверка корректности самого файла подписи.

Если во время проверки сертификат подписчика не был найден в локальном хранилище, то этот сертификат берется из подписываемого файла и устанавливается автоматически в локальное хранилище.

Общая схема выполнения процедуры проверки УЭП и схема соответствующих импортируемых функций криптоплатформы, которые нужно вызвать для реализации того или иного действия, приведены на рисунке 1.3.

Рисунок 1.3. Схема операции проверки УЭП

1.1.4 Шифрование файла

Шифрование производится на открытом ключе, содержащемся в сертификате. Закрытый ключ есть только у владельца использованного сертификата открытого ключа. Таким образом, при шифровании файла никто, кроме владельца закрытого ключа, не сможет расшифровать файл.

Криптопалтформа может производить шифрование файла сразу для нескольких будущих получателей файла, при этом их сертификаты должны быть созданы с помощью криптографического алгоритма, относящегося к единому стандарту. Для каждого сертификата получателей пользователь может просмотреть статус, чтобы на его основании сделать вывод о пригодности данного сертификата к шифрованию.

Также благодаря возможности гарантированного удаления файла после шифрования есть возможность сохранения полной секретности содержимого документа.

Общая схема выполнения процедуры шифрования файла и схема соответствующих импортируемых функций криптоплатформы, которые нужно вызвать для реализации того или иного действия, приведены на рисунке 1.4.

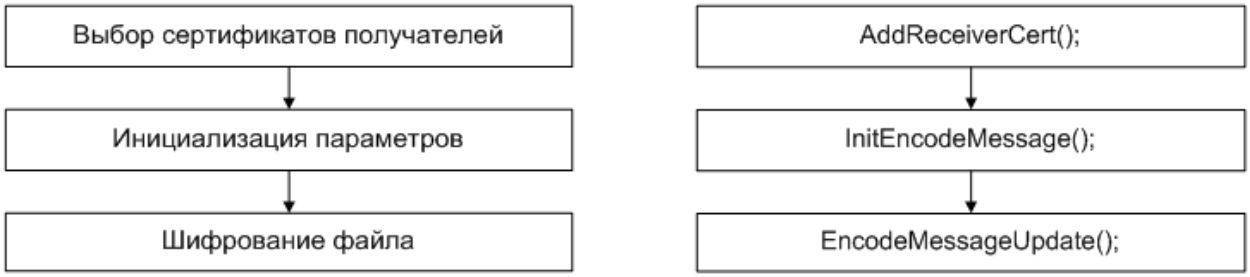


Рисунок 1.4. Схема операции шифрования файла

1.1.5 Расшифрование файла

При получении зашифрованного документа, если есть закрытый ключ, связанный с одним из открытых ключей, на которых производилось шифрование файла, то расшифровывание пройдет успешно. Если у пользователя несколько закрытых ключей, которым соответствуют несколько открытых ключей, участвующих при шифровании, то расшифровывание произойдет на первом из закрытых ключей. После расшифровывания пользователь может получить информацию о том, на каком сертификате была произведена операция расшифровывания.

Общая схема выполнения процедуры расшифровывания файла и схема соответствующих импортируемых функций криптоплатформы, которые нужно вызвать для реализации того или иного действия, приведены на рисунке 1.5.

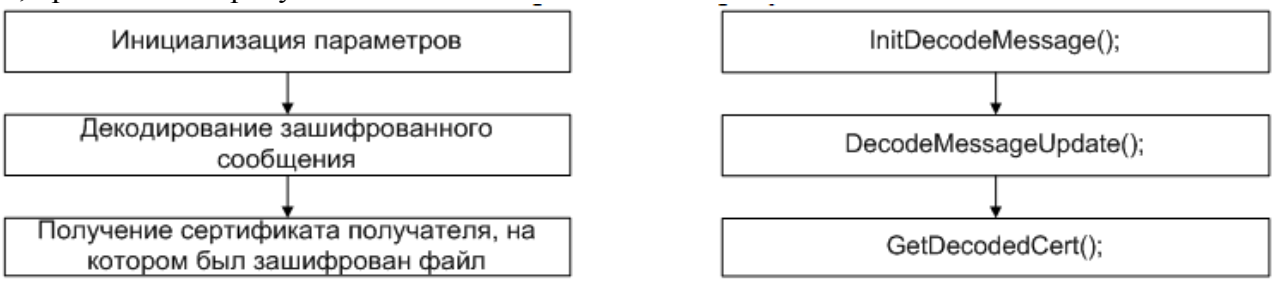


Рисунок 1.5. Схема операции расшифровывания файла

1.1.6 Сервисы системы ДТС

Система ДТС предназначена для обеспечения функционала подтверждения подписей, выполненных с использованием сертификатов, выданных различными (в том числе иностранными) удостоверяющими центрами.

Система интегрируется в инфраструктуру открытых ключей (ИОК) компании и используется ее клиентами, подразделениями и дочерними предприятиями при осуществлении бизнес-процессов, связанных с подтверждением подписи в электронных документах, защитой информации, обеспечением юридической силы и архивного хранения электронных документов.

Система ДТС предназначена для выполнения следующих функций:

- подтверждение ЭП сертификата, выданного доверенным удостоверяющим центром (список доверенных удостоверяющих центров определяется наличием соглашений о взаимном доверии с использующимся удостоверяющим центром, а также законодательством РФ);
- удостоверение обладания информацией в указанный момент времени;
- удостоверение обладания информацией без предоставления ее сервису (по хеш);
- подтверждение ЭП электронного документа.

Результатом выполнения той или иной функции ДТС служит созданная DVCS-сервером квитанция. Схема создания запроса на проверку подписи и предоставления отчета о проверке представлена на рисунке 1.6.

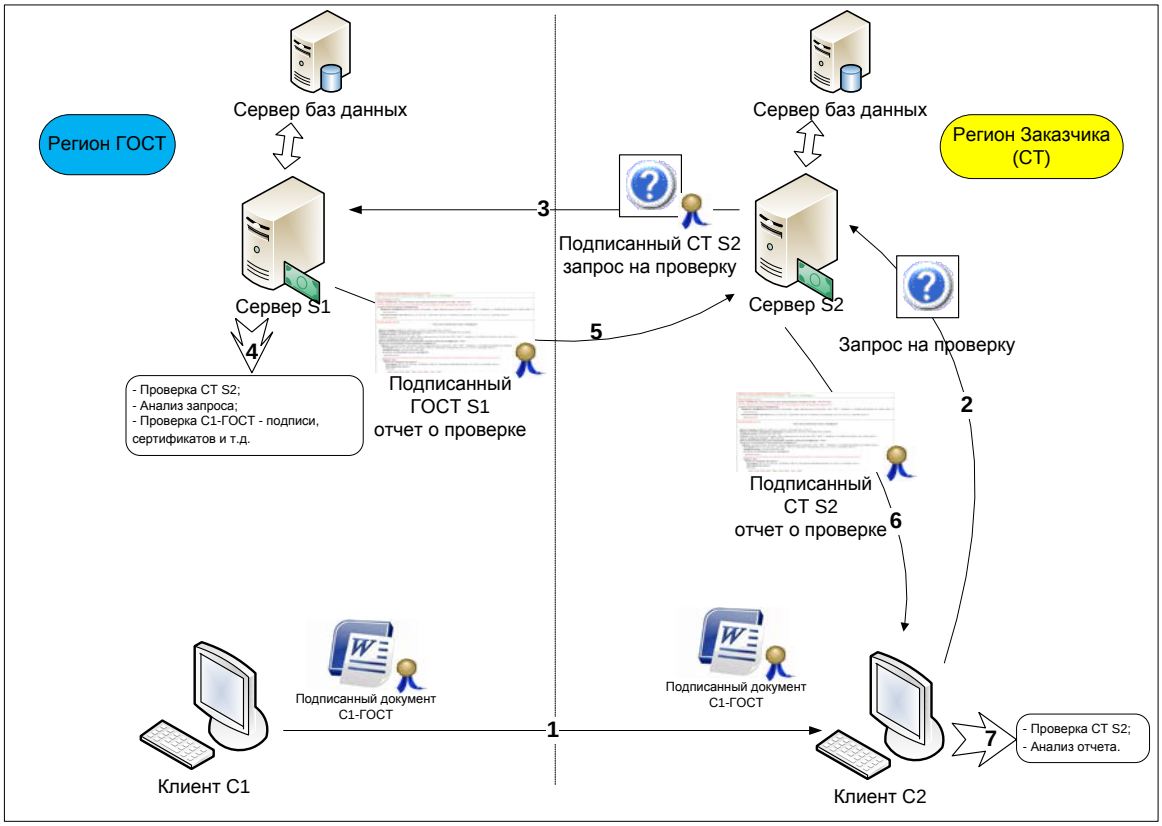


Рисунок 1.6. Схема запроса в инфраструктуре DVCS

Для осуществления проверки подписи, создания запроса на проверку и отчета о проверке на Клиентах С1, С2 и Серверах S1, S2 устанавливается криптографическая платформа «Litoria Crypto Platform».

В данном решении криптоплатформа (в соответствии с RFC3029) выполняет следующие функции:

- создание и отправка 4-х типов DVCS-запросов (CPD, CCPD, VSD, VPKC);
- формирование DVC-квитанции;
- анализ DVC-квитанции и вывод информации.

1.2 Требования к техническим средствам

Минимальные требования к рабочей станции, на которую устанавливается криптографическая платформа «Litoria Crypto Platform», обусловлены применением ОС: процессор с тактовой частотой 1 ГГц, RAM 2 Гб, HDD 20Гб, видеоадаптер SVGA, свободный USB-порт при использовании электронных идентификаторов или других USB-устройств для хранения цифровых сертификатов.

1.3 Требования к программному обеспечению

Криптоплатформа «Litoria Crypto Platform» функционирует под управлением следующих ОС:

- MS Windows 2000/XP/2003/Vista/2008/7/2008R2/8/2012/8.1/2012R2 (32- и 64-bit);
- Linux, удовлетворяющие стандарту LSB 4.x/3.x;
- Mac OS X (x64);
- iOS версии 4.2 и выше.

Файл криптоплатформы для различных ОС именуются согласно таблице 1.1.

Таблица 1.1. Наименования файла криптоплатформы

ОС	Имя файла криптоплатформы
Microsoft Windows	BHCryptography.dll
LSB	libBHCryptography.so
Mac OS X	libBHCryptography.dylib
iOS	libiPadEDS.a

Для работы криптоплатформы требуется установка криптопровайдера, реализованного в соответствии с технологией Microsoft CSP, а также установка соответствующего драйвера отчуждённого носителя. В качестве отчуждённого носителя может использоваться любой носитель, предусмотренный для использования установленным крипторовайдером.

1.4 Общая характеристика входной и выходной информации

1.4.1 Входные данные

Криптографическая платформа «Litoria Crypto Platform» выполняет функции шифрования/расшифровывания и создания/добавления/проверки УЭП для файлов произвольного типа.

1.4.2 Выходные данные

На выходе криптоплатформа «Litoria Crypto Platform» может формировать следующую информацию:

- байтовый массив данных (обработанный оригинал файла с ЭП);
- байтовый массив данных (файлы отделенных ЭП);
- сертификаты (из ЭП);
- четыре статуса проверок валидности подписи (ЭП на документ, ЭП на внутреннем штампе времени, ЭП на OCSP ответе, ЭП на внешнем штампе времени);
- дату и время из штампа времени;
- сообщение об ошибке (если она имеет место).

2 Структура криптоплатформы

Криптографическая платформа «Litoria Crypto Platform» реализована в виде динамической платформы на языках программирования C и C++. Классификация импортируемых функций и классов, содержащихся в исходных кодах криптоплатформы, приведена в таблице 2.1. Подробное описание функций и классов дано в соответствующих разделах.

Таблица 2.1. Классификация импортируемых функций и классов

Название группы	Функция/Класс	Назначение	Расположение
Работа с сертификата-ми	CreateNewKeys()	Создание ключевой пары и запроса на сертификат	CreateCertRequest.h
	CreateNewKeysUsingCert()	Создание ключевой пары и запроса на сертификат с использованием уже существующего сертификата	
	SetPrivateKeyAndInstallCert()	Установка сертификата в хранилище личные, в контейнер, а также создание связки открытый-закрытый ключ	
	класс CertRequest	Создание запроса на сертификат и установка полученного сертификата в контейнер	
	InstallCertToStore()	Установка сертификата в заданное хранилище	CertInfo.h
	GetCertInStore()	Получение сертификата из хранилища	
	GetCountCertAndOpenStore()	Открытие хранилища сертификатов, получение числа сертификатов в хранилище	
	GetNextCert()	Последовательное получение сертификатов из хранилища	
	GetOIDByCertAndType()	Получение подходящего алгоритма по заданному сертификату и типу	
	класс CertInfo	Получение информации о сертификате	
	GetCRLByCert()	Формирование CRL по сертификату	CrlManagment.h
	класс CrlManagment	Работа с CRL	
	GetCryptoProvNames()	Получение имен всех криптопровайдеров в системе	CryptoProvInfo.h
	GetCryptoProvParam()	Получение параметров криптопровайдера	
	GetCertFromContainer()	Получение сертификата из контейнера ключей	
	класс CryptoProvParamClass	Получение информации о криптопровайдере	

Название группы	Функция/Класс	Назначение	Расположение
	ShowImportDialog()	Вызов стандартного диалогового окна операции импорта сертификата	WizardDialogs.h
	DeleteCertificate()	Вызов стандартного диалогового окна операции удаления сертификата из хранилища	
	ShowExportDialog()	Вызов стандартного диалогового окна операции экспорта сертификата	
Хеширование	InitHashFile()	Инициализация параметров для хеширования файла	HashingFile.h
	UpdateHashFile()	Добавление следующего блока хеширования данных (потоковое чтение информации)	
	GetResultHashFile()	Формирование результата хеширования	
	класс HashingFile	Хеширование данных	
Выработка ЭП	InitCreateEds()	Инициализация параметров создания ЭП	CreateEDS.h
	CryptMessageUpdate()	Добавление данных на обработку (потоковое чтение информации)	
	AddSigned()	Добавление ЭП в сообщение	
	CounterSigned()	Заверка ЭП в сообщении	
	класс CreateEDS: public BHGiSCryptography	Создание/добавление простой ЭП	
	UpgradeSignedToAdvanced()	Усовершенствование ЭП	AdvancedEDS.h
	AddSignatureTimeStamp()	Включение штампа времени на значение ЭП	
	класс AdvancedEDS	Усовершенствование ЭП	
	InitMsgConvertContext()	Инициализация параметров для соединения отделенной ЭП и данных	MsgConvertContext.h
	MsgConvertUpdate()	Добавление данных (потоковое чтение информации)	
	MsgConvertFinish()	Завершение операции соединения	
	класс MsgConvertContext: public BHGiSCryptography	Соединение отделенной ЭП и данных	
	SignHash()	Подписание хеш-значения и формирования структуры отделенной ЭП в соответствии со стандартом pkcs#7	SignHash.h

Название группы	Функция/Класс	Назначение	Расположение
	InitAddSignByHash()	Инициализация параметров для добавления ЭП, используя механизм удаленной ЭП хеш-значения от исходного сообщения	AddSignByHash.h
	AddSignByHashJoinComponents()	Соединение всех компонентов воедино в процессе добавления ЭП, используя механизм удаленной ЭП хеш-значения от исходного сообщения	
	класс AddSignByHash	Добавление ЭП через удаленную ЭП хеш-значения	
Проверка ЭП	GetEncodedFileType()	Определение типа сообщения (подписанный файл, зашифрованный файл и т.д.)	VerifyEDS.h
	IsSignDetached()	Определение типа ЭП (отделенная или нет)	
	IsSignDetachedByFileName()	Определение типа ЭП (отделенная или нет) по имени файла	
	InitVerifyEds()	Инициализация данных для проверки ЭП	
	CryptMsgVerifyUpdate()	Добавление данных на обработку (потоковое чтение информации)	
	VerifyEdsSignature()	Проверка математической корректности ЭП, формирование результата проверки, возвращение количества ЭП в сообщении	
	GetVerifyEdsResult()	Получение результата проверки ЭП	
	VerifyEdsFreeResource()	Очистка памяти	
	класс VerifyEDS: public BHGiSCryptography	Проверка ЭП	
	класс VerifyAdvanced	Проверка доказательств УЭП	VerifyAdvanced.h
	класс OperationResult	Формирование информации об ошибке	OperationResult.h
Шифрование	AddReceiverCert()	Добавление сертификата получателя	Encrypt.h
	InitEncodeMessage()	Инициализация параметров шифрования сообщения	
	EncodeMessageUpdate()	Добавление данных на обработку (потоковое чтение информации)	
	класс EncryptClass: public BHGiSCryptography	Шифрование данных	

Название группы	Функция/Класс	Назначение	Расположение
Расшифровы-вание	InitDecodeMessage()	Инициализация параметров расшифровывания	Decrypt.h
	DecodeMessageUpdate()	Добавление данных на обработку (потоковое чтение информации)	
	GetDecodedCert()	Получение сертификата получателя, на котором был зашифрован файл	
	класс DecryptClass: public BHGiSCryptography	Шифрование данных	
DVCS-сервис	SignDVCSData()	Подписание данных в протоколе DVCS	DVCCommon.h
	VerifyDVCSSignature()	Проверка подписи данных в протоколе DVCS	
	CreateDVCSRequest()	Создание DVCS-запроса	DVCRequest.h
	CreateHashVsdRequest()	Создание DVCS-запроса типа VSD с хешированием данных	
	класс DVCRequest	Создание DVCS-запроса	
	GetOIDsFromDVCSRequest()	Получение списка алгоритмов подписей/сертификатов, содержащихся в DVCS-запросе	AnalyzeRequestData .h
	CreateDVCSResponse()	Создание DVCS-ответа по запросу	DVCResponse.h
	CreateDVCSErrorNotice()	Создание структуры ошибочного DVCS-ответа	
	ChangeDVCSerial()	Замена серийного номера в DVCS-ответе при трансляции на другой сервер ДТС	
	класс DVCResponse	Создание DVCS-ответа	
	GetAttrFromSign()	Получение DVC-квитанции из атрибута подписи	SignAttrWork.h
	AddAttrToSign()	Добавление DVC-квитанции в подпись как неподписываемый атрибут	
	ParseDVCResponce()	Анализ DVC-квитанции	DVCResponseParser .h
	класс DVCResponceParsing		
Сопутствую-щие	GetOcspResponseByCert()	Формирование OCSP-ответа по сертификату	OcspManagment.h
	класс OcspManagment	Работа с OCSP	
	SecureDeleteFile()	Гарантированное удаление файла	SecureDelete.h
	класс SecureDeleteClass		
	SetProxyNameAndPassword()	Установка имени и пароля пользователя для доступа к прокси-серверу	HttpDownloader.h
	класс HttpDownloader	Работа с сетевыми обращениями	
	класс ParseUrl	Анализ адреса	ParseUrl.h
	класс TspManagment	Работа со штампом времени	TspManagment.h

Название группы	Функция/Класс	Назначение	Расположение
	Класс общих операций		BHGiSCryptography.h
	Класс общих вспомогательных функций		Common.h
Работа с ошибками	GetErrors()	Получение информации о возникшей ошибке v.1	BHGiSError.h
	GetBHErrorInfo()	Получение информации о возникшей ошибке v.2	
	GetBHGiSErrorString()	Получение описания ошибки	
	класс BHGiSErrorClass	Формирование информации об ошибке	
	класс ErrorRouting	Управление формированием информации об ошибке	ErrorRouting.h

Импортируемые функции криптоплатформы могут быть вызваны из любого интерфейсного модуля.

На рисунке 2.1 представлено дерево вызовов h-файлов криптоплатформы «Litoria Crypto Platform».

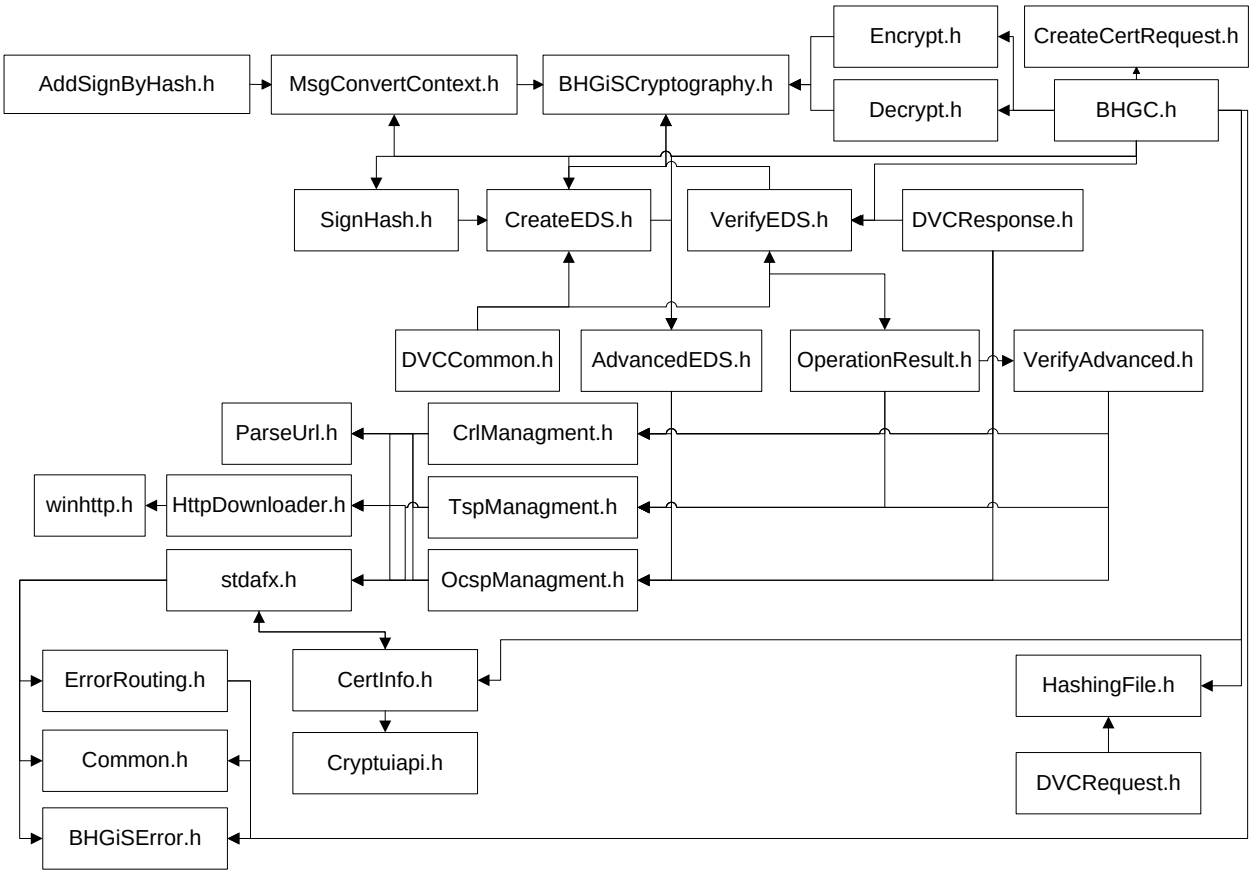


Рисунок 2.1. Дерево вызовов h-файлов криптоплатформы «Litoria Crypto Platform»

2.1 Работа с сертификатами

2.1.1 Функция CreateNewKeys()

Заголовок объявления функции:

BOOL __declspec(dllimport) CreateNewKeys(CertSubjectName_t certSubjectName, LPSTR* EnhKeyUsage, DWORD EnhKeyUsageCount, DWORD keyUsageValue, LPWSTR cryptoProvName, LPWSTR keyContainerName, BYTE* &pbCertRequest, DWORD& cbCertRequest);

Входные параметры:

certSubjectName – структура, описывающая информацию о владельце сертификата:

```
typedef struct CertSubjectName
{
    LPWSTR commonName;
    LPWSTR mail;
    LPWSTR organization;
    LPWSTR devision;
    LPWSTR title;
    LPWSTR town;
    LPWSTR region;
    LPWSTR country;
} CertSubjectName_t;
```

EnhKeyUsage – массив значений расширения сертификата «расширенное назначение ключа» (Extended Key Usage)

EnhKeyUsageCount – количество значений расширения Extended Key Usage

keyUsageValue – параметр для задания расширения «назначение ключа» (Key Usage)

cryptoProvName – имя используемого криптопровайдера

keyContainerName – имя контейнера ключевой пары

Выходные параметры:

pbCertRequest – сформированный запрос на сертификат в виде набора байт

cbCertRequest – размер запроса

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.2 Функция CreateNewKeysUsingCert()

Заголовок объявления функции:

BOOL __declspec(dllimport) CreateNewKeysUsingCert(BYTE* pbUsingCert, DWORD cbUsingCert, LPWSTR cryptoProvName, LPWSTR keyContainerName, BYTE* &pbCertRequest, DWORD& cbCertRequest);

Входные параметры:

pbUsingCert – используемый сертификат в виде набора байт

cbUsingCert – размер сертификата

cryptoProvName – имя используемого криптопровайдера

keyContainerName – имя контейнера ключевой пары

Выходные параметры:

pbCertRequest – сформированный запрос на сертификат в виде набора байт

cbCertRequest – размер запроса

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.3 Функция SetPrivateKeyAndInstallCert()

Заголовок объявления функции:

BOOL __declspec(dllimport) SetPrivateKeyAndInstallCert(BYTE* pbCertBlob, DWORD cbCertBlob, LPWSTR cryptoProvName, LPWSTR keyContainerName);

Входные параметры:

pbUsingCert – устанавливаемый сертификат в виде набора байт
cbUsingCert – размер сертификата
cryptoProvName – имя используемого криптопровайдера
keyContainerName – имя контейнера ключевой пары

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.4 Класс CertRequest

Переменные класса:

CertSubjectName_t *certSubjectName* – структура, описывающая информацию о владельце сертификата (содержимое структуры см. в п.п.2.1.1)
LPSTR* *enhKeyUsage* – массив расширений (Улучшенный ключ)
DWORD *enhKeyUsageCount* –количество расширений (Улучшенный ключ)
DWORD *keyUsageValue* – расширение Использование ключа
LPWSTR *cryptoProvName* – имя используемого криптопровайдера
LPWSTR *keyContainerName* – имя контейнера ключей
PCCERT_CONTEXT *usingCert* – используемый сертификат
DWORD *provType* – номер (тип) криптопровайдера
HCRYPTPROV *hCryptProv* – указатель на криптопровайдер
PCERT_PUBLIC_KEY_INFO *pbPublicKeyInfo* – открытый ключ
DWORD *cbPublicKeyInfo* – размер ключа
PCERT_NAME_BLOB *subjNameBlob* – информация о субъекте
PCRYPT_ATTRIBUTE *certRequestAttrib* – атрибуты запроса на сертификат
int *certRequestAttribCount* – количество атрибутов
LPSTR *base64Request* – запрос на сертификат, кодированный в base64

Внутренние функции:

DWORD GetProvType() – определение типа криптопровайдера
void SetKeyContainerName() – установка имени ключевого контейнера
bool FindKeyContainerName() – поиск ключевого контейнера
bool FinishInstall() – завершение установки сертификата
bool CreateKyes() – создание пары ключей, определение контекста криптопровайдера
PCERT_NAME_BLOB EncodeCertSubjNameBlob() – формирование информации о владельце сертификата
PCRYPT_ATTRIBUTE EncodeCertRequestAttrib() – формирование атрибутов запроса на сертификат
CRYPT_INTEGER_BLOB* OSVertion() – формирование атрибута – версия операционной системы
CRYPT_INTEGER_BLOB* SubjectKeyIdentifier() – формирование атрибута – идентификатор открытого ключа

bool CreateBase64CertRequest() – формирование запроса на сертификат
void FreeResourse() – очистка занятых ресурсов

Открытые функции:

1) Конструктор для создания запроса без использования сертификата

```
CertRequest(CertSubjectName_t certSubjectName, LPSTR* enhKeyUsage, DWORD enhKeyUsageCount, DWORD keyUsageValue, LPWSTR cryptoProvName, LPWSTR keyContainerName)
        :certSubjectName(certSubjectName),
        enhKeyUsage(enhKeyUsage),
        enhKeyUsageCount(enhKeyUsageCount),
        keyUsageValue(keyUsageValue),
        cryptoProvName(cryptoProvName),
        keyContainerName(keyContainerName),
        usingCert(nullptr),
        provType(0xFFFFFFFF),
        hCryptProv(0),
        pbPublicKeyInfo(nullptr),
        cbPublicKeyInfo(0),
        subjNameBlob(nullptr),
        certRequestAttrib(nullptr),
        certRequestAttribCount(0),
        base64Request(nullptr)
```

2) Конструктор для создания запроса с использованием сертификата

```
CertRequest(BYTE* pbUsingCertBlob, DWORD cbUsingCertBlob, LPWSTR cryptoProvName, LPWSTR keyContainerName)
        :certSubjectName(),
        enhKeyUsage(nullptr),
        enhKeyUsageCount(0),
        keyUsageValue(0),
        cryptoProvName(cryptoProvName),
        keyContainerName(keyContainerName),
        usingCert(nullptr),
        provType(0xFFFFFFFF),
        hCryptProv(0),
        pbPublicKeyInfo(nullptr),
        cbPublicKeyInfo(0),
        subjNameBlob(nullptr),
        certRequestAttrib(nullptr),
        certRequestAttribCount(0),
        base64Request(nullptr)
{
        usingCert = CertCreateCertificateContext(TYPE_DER,
        pbUsingCertBlob,
         cbUsingCertBlob);
}
```

3) Деструктор

```
~CertRequest()
{
        FreeResourse();
}
```

4) Создание пары ключей и запроса
bool CreateKeysAndCertRequest();
5) Установка сертификата
bool InstallCert();
6) Возврат запроса в виде массива байт
std::vector<BYTE> GetCertRequestData();

2.1.5 Функция InstallCertToStore()

Заголовок объявления функции:

BOOL __declspec(dllimport) InstallCertToStore(const BYTE* pbCertBlob, DWORD cbCertBlob, LPWSTR storeName);

Входные параметры:

pbCertBlob – устанавливаемый сертификат в виде набора байт
cbCertBlob – размер сертификата
storeName – имя хранилища сертификатов

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var CertRawData = certForInstall.RawData;
if (!BHImportFunctions.InstallCertToStore(CertRawData, (uint)CertRawData.Length, storeName))
{
    var edsException = new EDSException();
    edsException.SetBHError();
    throw (edsException);
}
```

2.1.6 Функция GetCertInStore()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetCertInStore(BYTE* &pbCertRawData, DWORD& cbCertRawData, LPWSTR storeName);

Входные параметры:

storeName – имя хранилища сертификатов

Выходные параметры:

pbCertRawData – выбранный сертификат в виде набора байт
cbCertRawData – размер сертификата

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.7 Функция GetCountCertAndOpenStore()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetCountCertAndOpenStore(LPWSTR storeName, DWORD& countCerts);

Входные параметры:

storeName – имя хранилища сертификатов

Выходные параметры:

countCerts – количество сертификатов в хранилище

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
uint size;
if (!BHImportFunctions.GetCountCertAndOpenStore(storeName, out size))
{
        var edsException = new EDSException();
        edsException.SetBHError();
        throw (edsException);
}
```

2.1.8 Функция GetNextCert()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetNextCert(BYTE* &pbNextCertRawData, DWORD& cbNextCertRawData);

Выходные параметры:

pbNextCertRawData – сертификат в виде набора байт
cbNextCertRawData – размер сертификата

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
uint size;
for (int i = 0; i < size; i++)
{
        uint certRawDataSize;
        var certRawDataPtr = IntPtr.Zero;
        if (!BHImportFunctions.GetNextCert(ref certRawDataPtr, out certRawDataSize))
        {
                var edsException = new EDSException();
                edsException.SetBHError();
                throw (edsException);
        }
        var certRawData = new byte[certRawDataSize];
        Marshal.Copy(certRawDataPtr, certRawData, 0, (int)certRawDataSize);
        BHImportFunctions.FreePtrMemory(certRawDataPtr);
        collection.Add(new X509Certificate2(certRawData));
}
```

2.1.9 Функция GetOIDByCertAndType()

Заголовок объявления функции:

LPWSTR __declspec(dllimport) GetOIDByCertAndType(const BYTE* pbCertRawData, DWORD cbCertRawData, DWORD algType);

Входные параметры:

cbCertRawData – размер сертификата
algType – тип криптографического алгоритма

Выходные параметры:

pbCertRawData – выбранный сертификат в виде набора байт

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение – OID подходящего алгоритма.
Если функция завершается ошибкой, возвращаемое значение – NULL.
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.10 Класс CertInfo

Структура для получения сертификатов из хранилища:

```
typedef struct GetCertInfo
{
	HCERTSTORE hGetCertStore;
	PCCERT_CONTEXT nextCert;
public:
	GetCertInfo()
	{
		hGetCertStore = nullptr;
		nextCert = nullptr;
	}
	~GetCertInfo()
	{
		if(hGetCertStore)
			CertCloseStore(hGetCertStore, 0);
		if(nextCert)
			CertFreeCertificateContext(nextCert);
	}
} GetCertInfo_t;
```

Открытые функции:

1) Получение точек распространения OCSP для заданного сертификата
static bool GetOCSPUrls(PCCERT_CONTEXT pCertContext, BSTR*& ocspAdress, DWORD &countAdress);
2) Получение точек распространения CRL для заданного сертификата
static bool GetCRLUrls(PCCERT_CONTEXT pCertContext, BSTR*& crlAdress, DWORD &countAdress);
3) Получение криптографического алгоритма по типу и сертификату
static bool GetAvailableCryptoprovAlgsByCert(PCCERT_CONTEXT pCertContext, DWORD algType, ALG_ID& algId, HCRYPTPROV &hProv);
4) Получение криптографического алгоритма по типу и дескриптору криптопровайдера
static bool GetAvailableCryptoprovAlgsByHProv(HCRYPTPROV hProv, DWORD algType, ALG_ID& algId);
5) Получение сертификата заданного поставщика
static PCCERT_CONTEXT GetIssuerCert(PCCERT_CONTEXT pCert);
6) Формирование цепочки сертификации
static PCCERT_CHAIN_CONTEXT GetCertificateChain(PCCERT_CONTEXT pCertContext);
7) Формирование цепочки сертификации с использованием дополнительного хранилища сертификатов
static PCCERT_CHAIN_CONTEXT GetCertificateChainWithAddStore (PCCERT_CONTEXT pCertContext, HCERTSTORE hAddStore, FILETIME verifyTSTime);

2.1.11 Функция GetCRLByCert()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetCRLByCert(BYTE* pbCert, DWORD cbCert, BOOL download, BOOL install, BYTE*& pbCRL, DWORD &cbCRL);

Входные параметры:

pbCert – сертификат в виде набора байт
cbCert – размер сертификата
download – флаг загрузки CRL из сети
install – флаг установки CRL в хранилище

Выходные параметры:

pbCRL – CRL в виде набора байт
cbCRL – размер CRL

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.12 Класс CrlManagment

Открытые функции:

1) Получение CRL из хранилища сертификатов
static PCCRL_CONTEXT GetLocalCRLByCert(PCCERT_CONTEXT certToCheck);
2) Получение CRL из сети
static PCCRL_CONTEXT DownloadCrl(PCCERT_CONTEXT certToCheck, BOOL install);
3) Проверка сертификата на отозванность по заданному CRL
static bool IsCertRevokedByCrl(PCCERT_CONTEXT certToCheck, PCCRL_CONTEXT crl);
4) Формирование контекста CRL из массива байт
static bool CreateCrlContext(std::vector<BYTE> response, PCCRL_CONTEXT &pCrl);

2.1.13 Функция GetCryptoProvNames()

Заголовок объявления функции:

int __declspec(dllimport) GetCryptoProvNames(wchar_t** &cryptoProvNames);

Выходные параметры:

cryptoProvNames – массив имен криптопровайдеров

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение – количество криптопровайдеров в системе.
Если функция завершается ошибкой, возвращаемое значение равно 0.
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.14 Функция GetCryptoProvParam()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetCryptoProvParam(LPWSTR cryptoProvName, CryptoProvParam_t& provParam);

Входные параметры:

cryptoProvName – имя криптопровайдера
provParam – структура, описывающая параметры криптопровайдера

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.15 Функция GetCertFromContainer()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetCertFromContainer(LPWSTR cryptoProvName, DWORD cryptoProvType, LPWSTR keyContainerName, BYTE* &pbCertContext, DWORD& cbCertContext);

Входные параметры:

cryptoProvName – имя криптопровайдера
cryptoProvType – тип криптопровайдера
keyContainerName – имя контейнера ключей
pbCertContext – сертификат в контейнере ключей в виде набора байт
cbCertContext – размер сертификата

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.16 Класс CryptoProvParamClass

Переменные класса:

LPWSTR cryptoProvName – имя криптопровайдера
HCRYPTPROV hCryptProv – указатель на контекст криптопровайдера

Внутренние функции:

DWORD GetProvType() – определение типа криптопровайдера

Открытые функции:

1) Конструктор

```
CryptoProvParamClass(LPWSTR cryptoProvName)
    :cryptoProvName(cryptoProvName)
```

2) Возврат структуры параметров криптопровайдера

```
bool GetProvParam(CryptoProvParam_t& cryptoProvParam);
```

Структура параметров криптопровайдера:

```
typedef struct CryptoProvParam
{
    DWORD provType;
    DWORD containerCount;
    wchar_t** containerNames;
    DWORD algCount;
    PROV_ENUMALGS_EX** algInfo;
} CryptoProvParam_t;
```

2.1.17 Функция ShowImportDialog()

Заголовок объявления функции:

BOOL __declspec(dllimport) ShowImportDialog(DWORD parent, LPWSTR title);

Входные параметры:

parent – дескриптор родительского окна
title – заголовок окна

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.18 Функция DeleteCertificate()

Заголовок объявления функции:

BOOL __declspec(dllimport) DeleteCertificate(LPWSTR storeName, BYTE *pbCertEncoded, DWORD cbCertEncoded);

Входные параметры:

storeName – имя хранилища сертификатов
pbCertEncoded – сертификат для удаления в виде набора байт
cbCertEncoded – размер сертификата

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.1.19 Функция ShowExportDialog()

Заголовок объявления функции:

BOOL __declspec(dllimport) ShowExportDialog(DWORD parent, LPWSTR title, BYTE *pbCertEncoded, DWORD cbCertEncoded);

Входные параметры:

parent – дескриптор родительского окна
title – заголовок окна
pbCertEncoded – сертификат для экспорта в виде набора байт
cbCertEncoded – размер сертификата

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.2 Хеширование

2.2.1 Функция InitHashFile()

Заголовок объявления функции:

BOOL __declspec(dllimport) InitHashFile(LPSTR hashOid);

Входные параметры:

hashOid – идентификатор алгоритма хеширования

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.2.2 Функция UpdateHashFile()

Заголовок объявления функции:

BOOL __declspec(dllimport) UpdateHashFile(BYTE* pbContent, DWORD cbContent);

Входные параметры:

pbContent – блок данных для хеширования
cbContent – размер блока данных

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.2.3 Функция GetResultHashFile()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetResultHashFile(BYTE* &pbFileHash, DWORD& cbFileHash);

Входные параметры:

cbFileHash – размер хеш-значения

Выходные параметры:

pbFileHash – рассчитанное хеш-значение

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.2.4 Класс HashingFile

Переменные класса:

HCRYPTHASH hHashFile – дескриптор
LPSTR hashOid – алгоритм хеширования

Открытые функции:

1) Конструктор

```
HashingFile(LPSTR hashOid)
        :hashOid(hashOid),
        hHashFile(0)
```

2) Деструктор

```
~HashingFile()
{
        CryptDestroyHash(hHashFile);
}
```

3) Инициализация параметров

```
bool InitHashParam();
```

4) Обновление дескриптора новым блоком данных

```
bool CryptHashUpdate(BYTE* pbContent, DWORD cbContent);
```

5) Возврат сформированного хеш-значения

```
std::vector<BYTE> GetHashResult();
```

7) Возврат алгоритма хеширования

```
LPSTR GetHashOid();
```

2.3 Выработка ЭП

2.3.1 Функция InitCreateEds()

Заголовок объявления функции:

BOOL __declspec(dllimport) InitCreateEds(BYTE* pbSignerCert, DWORD cbSignerCert, LPWSTR outFileName, BYTE* comment, DWORD createEdsFlags);

Входные параметры:

pbSignerCert – сертификат в виде набора байт
cbSignerCert – размер сертификата
outFileName – имя выходного файла
comment – комментарий к ЭП
createEdsFlags – флаги создания ЭП (Таблица 2.2 раздела 2.3.4)

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var CertRawData = certificate.RawData;
EdsFlags = 0;
EdsFlags = CreateEdsFlags.INCLUDE_TIME_STAMP;
var signatureComment = "Demo-версия GiSCryptoNet";
var comment = Encoding.Unicode.GetBytes(signatureComment);
using (var inputStream = new System.IO.FileStream(inFileName, System.IO.FileMode.Open,
System.IO.FileAccess.Read, System.IO.FileShare.ReadWrite))
{
    if (!BHImportFunctions.InitCreateEds(CertRawData, (uint)CertRawData.Length,
    outputFileName, comment, (uint)EdsFlags))
    {
        var edsException = new EDSException();
        edsException.SetBHError();
        throw (edsException);
    }
```

2.3.2 Функция CryptMessageUpdate()

Заголовок объявления функции:

```
BOOL __declspec(dllimport) CryptMessageUpdate(BYTE* pbData, DWORD cbData, BOOL
lastCall);
```

Входные параметры:

pbData – блок данных в виде набора байт
cbData – размер блока данных
lastCall – флаг конца данных

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var LastCall = 0;
var buffer = new byte[SizeBuff];
int bytesRead;
while ((bytesRead = inputStream.Read(buffer, 0, SizeBuff)) > 0)
{if (bytesRead < SizeBuff)
    {
        LastCall = 1;}
        if (!BHImportFunctions.CryptMessageUpdate(buffer, (uint)bytesRead, LastCall))
    {
        var edsException = new EDSException();
        edsException.SetBHError();
        throw (edsException);
    }
}if (LastCall == 0)
{
    LastCall = 1;
    if (!BHImportFunctions.CryptMessageUpdate(buffer, (uint)0, LastCall))
    {
        var edsException = new EDSException();
        edsException.SetBHError();
        throw (edsException);}
}
```

2.3.3 Функция AddSigned()

Заголовок объявления функции:

BOOL __declspec(dllimport) AddSigned(BYTE* &pbEncodedBlob, DWORD& cbEncodedBlob, DWORD& newSignedIndex);

Выходные параметры:

pbEncodedBlob – подписанное сообщение, содержащее вновь добавленную подпись
cbEncodedBlob – размер сообщения
newSignedIndex – индекс вновь добавленной подписи в сообщение

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var signedDataPtr = IntPtr.Zero;
uint signedSize;
uint signatureIndex;
if (!BHImportFunctions.AddSigned(ref signedDataPtr, out signedSize, out signatureIndex))
{
        var edsException = new EDSException();
        edsException.SetBHError();
        throw (edsException);
}
```

2.3.4 Функция CounterSigned()

Заголовок объявления функции:

BOOL __declspec(dllimport) CounterSigned(DWORD signedIndex, BOOL isCades, BSTR tspAdress, BYTE* &pbEncodedBlob, DWORD& cbEncodedBlob);

Входные параметры:

signedIndex – индекс заверяемой подписи в сообщении
isCades – флаг усовершенствованной подписи
tspAdress – адрес службы штампов времени

Выходные параметры:

pbEncodedBlob – подписанное сообщение, содержащее вновь заверенную подпись
cbEncodedBlob – размер сообщения

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.3.5 Класс CreateEDS: public BHGiSCryptography

Переменные класса:

bool detachedEds – создание отделенной ЭП
bool *includeTimeStamp* – включение штампа времени
bool *coordinationPut* – согласование
bool *addEds* – добавление подписи
bool *dvcsFlag* – создание ЭП данных протокола DVCS
LPWSTR *outFileName* – имя файла для вывода преобразованных данных
PCCERT_CONTEXT *signerCert* – контекст сертификата
HCRYPTPROV *hProv* – контекст криптопровайдера
DWORD *keyType* – информация о ключе
ALG_ID *algId* – алгоритм хеширования
CRYPT_ATTRIBUTE *ca[3]* – подписываемые атрибуты подписи
CERT_BLOB *SignerCertBlob[1];* CMSG_SIGNER_ENCODE_INFO *signerEncodeInfo[1]* – структуры, содержащаие параметры создания/добавления ЭП
std::vector<BYTE> *signerCertRef* – ссылка на значение сертификата подписчика в виде массива байт
std::vector<std::vector<BYTE>> *existingSigns* – значения ЭП, присутствующие в сообщении до добавления новой подписи

Внутренние функции:

void ParseFlags(DWORD createEdsFlags) – анализ флагов создания ЭП

Таблица 2.2. Флаги параметров создания ЭП

Флаг	**Маска**	**Описание**
DETACHED_EDS	0x00000001	Тип ЭП – отделенная
INCLUDE_TIME_STAMP	0x00000002	Включение штампов времени вЭП
COORDINATION_PUT	0x00000004	Создание заверяющей ЭП
ADD_EDS	0x00000008	Добавление ЭП
DVCS_SIGNATURE	0x00000010	Подпись данных в протоколе DVCS

bool CreateCertRefV2() – создание ссылки на сертификат подписчика для сертификатов, отличных от Sha-1
bool CreateCertRefV1() – создание ссылки на сертификат подписчика для сертификатов Sha-1
CRYPT_INTEGER_BLOB* SignTime() – формирование атрибута времени подписания
CRYPT_INTEGER_BLOB* Comment(BYTE *comment) – формирование атрибута комментария к подписи
bool GetExisitingSigns() – получение значений подписей, присутствующих в сообщении
bool GetNewSignerIndex(std::vector<BYTE> signature, long &newSignatureIndex) – получение индекса вновь добавленной подписи
bool FindExisitingDigestOid() – определение присутствия необходимого поддерживаемого алгоритма хеширования в сообщении; добавление алгоритма в set в случае необходимости
bool AddUnsignedAttribute(long signatureIndex, std::vector<BYTE> attrData, LPSTR oid) – добавление неподписываемого атрибута в сообщение (используется для заверки подписи)
void FreeResourse() – освобождение занимаемых ресурсов

Открытые функции:

1) Конструктор

```
CreateEDS(LPWSTR outFileName, DWORD createEdsFlags)
        :detachedEds(false),
        includeTimeStamp(false),
        coordinationPut(false),
        addEds(false),
        dvcsFlag(false),
        outFileName(outFileName),
        signerCert(NULL),
        hProv(0),
        keyType(0),
        algId(0),
        signerCertRef(0)
{
        ParseFlags(createEdsFlags);
}
```

2) Деструктор

```
~CreateEDS()
{
        FreeResourse();
}
```

3) Инициализация криптографических параметров создания подписи

```
bool InitCreateEdsParam(BYTE* pbSignerCert, DWORD cbSignerCert, LPSTR pinCode = nullptr);
```

4) Инициализация дескриптора сообщения для создания подписи

```
bool InitCreateEdsHMsg(BYTE* comment, LPSTR contentObjID = NULL);
```

5) Инициализация дескриптора сообщения для "сложной" (УЭП, TSP) заверки подписи

```
bool InitCreateEdsHMsgForCoordinate(CMSG_SIGNER_ENCODE_INFO signerEncodeInfo[1]);
```

6) Флаг создания ЭП (не добавление, не заверка)

```
bool isCreateEdsOperation();
```

7) Добавление новой подписи в сообщение

```
bool AddNewSignature(std::vector<BYTE> &signature, long& newSignatureIndex);
```

8) Заверка подписи в сообщении

```
bool CoordinateSignature(long SignatureIndex,bool isCades, BSTR tspAdress, std::vector<BYTE> &signature);
```

9) Создание простой подписи

```
bool SignedSimpleMessage(BYTE* comment, std::vector<BYTE>& signedEncode);
```

2.3.6 Функция UpgradeSignedToAdvanced()

Заголовок объявления функции:

```
BOOL __declspec(dllimport) UpgradeSignedToAdvanced(BYTE* pbSigned, DWORD cbSigned,
DWORD signerIndex, BSTR tspAdress, BYTE*& pbAdvance, DWORD& cbAdvance);
```

Входные параметры:

pbSigned – исходное подписанное сообщение, содержащее отделенную подпись для усовершенствования в виде набора байт

cbSigned – размер исходного сообщения

signerIndex – индекс подписи в сообщении

tspAdress – адрес службы штампов времени

Выходные параметры:

pbAdvance – результат операции – подписанное сообщение, содержащее отделенную УЭП в виде набора байт

cbAdvance – размер сформированного сообщения

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var buffer = new byte[SizeBuff];
uint advanceSize;
var advancePtr = IntPtr.Zero;
if (!BHImportFunctions.UpgradeSignedToAdvanced(buffer, (uint)buffer.Length, 0, tspUrl, ref advancePtr, out advanceSize))
{
    var edsException = new EDSException();
    edsException.SetBHError();
    throw (edsException);
}
```

2.3.7 Функция AddSignatureTimeStamp()

Заголовок объявления функции:

BOOL __declspec(dllimport) AddSignatureTimeStamp(BYTE* pbSigned, DWORD cbSigned, DWORD signerIndex, BSTR tspAdress, BYTE*& pbAdvance, DWORD& cbAdvance);

Входные параметры:

pbSigned – исходное подписанное сообщение, содержащее отделенную подпись для усовершенствования в виде набора байт
cbSigned – размер исходного сообщения
signerIndex – индекс подписи в сообщении
tspAdress – адрес службы штампов времени

Выходные параметры:

pbAdvance – результат операции – подписанное сообщение, содержащее отделенную подпись со штампом времени в виде набора байт
cbAdvance – размер сформированного сообщения

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.3.8 Класс AdvancedEDS

Переменные класса:

PCCERT_CONTEXT pSignerCert – контекст сертификата подписчика
PCCERT_CHAIN_CONTEXT *pSignerChainContext* – цепочка сертификата подписчика
PCCERT_CONTEXT *pTspCert* – контекст сертификата tsp оператора
PCCERT_CHAIN_CONTEXT *pTspCertChainContext* – цепочка сертификата tsp оператора
HCRYPTMSG *hMsgAdvanced* – дескриптор сообщения для усовершенствования ЭП
BYTE* *pbSigned* – исходная отделенная подпись
DWORD *cbSigned* – размер исходной подписи
ALG_ID *algid* – алгоритм хеширования
HCRYPTPROV *hProv* – контекст криптопровайдера
DWORD *signedIndex* – индекс обрабатываемой подписи в сообщении
BSTR *tspAdress* – адрес используемой службы штампов времени
std::vector<PCCERT_CONTEXT> *_certificateSignerValues* – вектор сертификатов цепочки подписчика

std::vector<PCCERT_CONTEXT> *_certificateTSPValues* – вектор сертификатов цепочки tsp
std::vector<PCCERT_CONTEXT> *_certificateOcspValues*; std::vector<PCCERT_CONTEXT> *_certificateAllOcspValues*; – вектора сертификатов ocsp
std::vector<BYTE> *signBlob* – значение ЭП
TimeStampToken_t **signatureTimeStamp* – внутренний штамп времени
TimeStampToken_t **cadesTimeStamp* – внешний штамп времени
RevocationValues_t* *revValues* – структура, содержащая значения отзывов сертификатов
CompleteRevocationRefs_t* *completeRevocationRefs* – структура, содержащая ссылки на значения отзывов сертификатов
CertificateValues_t* *certificateValues* – структура, содержащая значения сертификатов
CompleteCertificateRefs_t* *completeCertificateRefs* – структура, содержащая ссылки значения сертификатов
FILETIME *tspInFileTime* – время формирования внутреннего штампа
FILETIME *tspOutFileTime* – время формирования внешнего штампа
std::vector<FILETIME> *ocspTimes* – вектор времен ocsp ответов, необходимый для проверки на рассинхронизацию со временем службы tsp

Внутренние функции:

bool SetAdvancedParam() – установка начальных параметров
bool CreateRevocationValuesAndRefs() – формирование атрибутов – значения отзывов, ссылки на значения отзывов
bool RevocationValuesAndRefsForChain(std::vector<PCCERT_CONTEXT> certs, DWORD startCert) – формирование атрибутов – значения отзывов, ссылки на значения отзывов для цепочки сертификатов
bool CreateCertificateValuesAndRefs() – формирование атрибутов – значения сертификатов, ссылки на значения сертификатов
bool CertificateValuesAndRefsForChain(std::vector<PCCERT_CONTEXT> certs, DWORD startCert) – формирование атрибутов – значения сертификатов, ссылки на значения сертификатов для цепочки сертификатов
TimeStampToken_t* GetTimeStamp(std::vector<BYTE> dataForTimeStamping, FILETIME& tspFileTime, PCCERT_CONTEXT& tspCert) – формирование штампа времени
TimeStampToken_t* GetSignatureTimeStamp(std::vector<BYTE> dataForTimeStamping) – формирование штампа времени в случае простой ЭП
std::vector<BYTE> GetDataToExternalTSPRequest() – формирование данных для внешнего штампа времени
bool AddAttributeToSign() – добавление атрибутов в подпись
bool AddUnsignedAttribute(std::vector<BYTE> attrData, LPSTR oid) – добавление неподписанного атрибута в подпись
bool CreateCadesCheckProofsValidity() – проверка действительности сертификатов на момент получения внешнего штампа времени
void FreeResourse() – освобождение занятой памяти

Открытые функции:

1) Конструктор

```
AdvancedEDS(BYTE* pbSigned, DWORD cbSigned, DWORD signedIndex, BSTR tspAdress)
    :pSignerCert(nullptr),
    pSignerChainContext(nullptr),
    pTspCert(nullptr),
    pTspCertChainContext(nullptr),
    hMsgAdvanced(nullptr),
    pbSigned(pbSigned),
    cbSigned(cbSigned),
    algId(0),
    hProv(0),
    signedIndex(signedIndex),
    tspAdress(tspAdress),
    signatureTimeStamp(nullptr),
    cadesTimeStamp(nullptr),
    revValues(nullptr),
    completeRevocationRefs(nullptr),
    certificateValues(nullptr),
    completeCertificateRefs(nullptr)
```

2) Деструктор

```
~AdvancedEDS()
{FreeResourse();}
```

3) Формирование усовершенствованной подписи

```
bool AdvancedSignature();
```

4) Формирование штампа времени на значение сигнатуры и добавление штампа в подписанный файл

```
bool TimeStampingSignature();
```

5) Формирование результата

```
std::vector<BYTE> GetAdvancedResult();
```

2.3.9 Функция InitMsgConvertContext()

Заголовок объявления функции:

```
BOOL __declspec(dllimport) InitMsgConvertContext(BSTR outFileName, BYTE* pbAdvance,
DWORD cbAdvance);
```

Входные параметры:

outFileName – имя результирующего файла

pbAdvance – отделенная УЭП в виде набора байт

cbAdvance – размер УЭП

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).

Если функция завершается ошибкой, возвращаемое значение равно нулю (false).

Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var advanceData = new byte[advanceSize];
Marshal.Copy(advancePtr, advanceData, 0, (int)advanceSize);
BHImportFunctions.FreePtrMemory(advancePtr);
if (!BHImportFunctions.InitMsgConvertContext(outputFileName, advanceData, advanceSize))
{var edsException = new EDSException();
    edsException.SetBHError();
    throw (edsException);}
```

2.3.10 Функция MsgConvertUpdate()

Заголовок объявления функции:

BOOL __declspec(dllimport) MsgConvertUpdate(BYTE* pbData, DWORD cbData, BOOL lastCall);

Входные параметры:

pbData – блок данных в виде набора байт
cbData – размер блока данных
lastCall – флаг конца данных

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
LastCall = 0;
buffer = new byte[SizeBuff];
while ((bytesRead = inputStream.Read(buffer, 0, SizeBuff)) > 0)
{
	if (bytesRead < SizeBuff)
	{
		LastCall = 1;
	}
	if (!BHImportFunctions.MsgConvertUpdate(buffer, (uint)bytesRead, LastCall))
	{
		var edsException = new EDSException();
		edsException.SetBHError();
		throw (edsException);
	}
}
```

2.3.11 Функция MsgConvertFinish()

Заголовок объявления функции:

BOOL __declspec(dllimport) MsgConvertFinish();

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
if (!BHImportFunctions.MsgConvertFinish())
{
	var edsException = new EDSException();
	edsException.SetBHError();
	throw (edsException);
}
```

2.3.12 Класс MsgConvertContext: public BHGiSCryptography

Переменные класса:

BSTR outFileName – файл для вывода данных

std::vector<BYTE> advansedSignature – отделенная подпись

SignedData_t *signedData – структура signedData

Внутренние функции:

void FreeResourse() – освобождение занятых ресурсов

Открытые функции:

1) Конструктор

```
MsgConvertContext(BSTR outFileName, BYTE* pbAdvance, DWORD cbAdvance)
        :outFileName(outFileName),
        signedData(0)
{
        advansedSignature.reserve(cbAdvance);
        for(int i = 0; i < cbAdvance; i++)
                advansedSignature.push_back(pbAdvance[i]);
}
```

2) Деструктор

```
~MsgConvertContext()
{
        FreeResourse();
}
```

3) Инициализация параметров конвертирования

```
bool InitConvertParam();
```

4) Завершение операции конвертирования

```
bool FinishConvert();
```

2.3.13 Функция SignHash()

Заголовок объявления функции:

```
BOOL __declspec(dllimport) SignHash(BYTE* pbHash, DWORD cbHash, BYTE*
pbSignerCert, DWORD cbSignerCert, BYTE* comment, BYTE* &pbResult, DWORD&
cbResult);
```

Входные параметры:

pbHash – подписываемое хеш-значение в виде набора байт

cbHash – размер хеш-значения

pbSignerCert – сертификат ключа подписи в виде набора байт

cbSignerCert – размер сертификата

comment – комментарий к подписи

Выходные параметры:

pbResult – результат операции – отделенная подпись в виде набора байт

cbResult – размер отделенной подписи

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.3.14 Функция InitAddSignByHash()

Заголовок объявления функции:

BOOL __declspec(dllimport) InitAddSignByHash(LPWSTR fileName, LPWSTR dataFileName, BYTE* pbSignerCert, DWORD cbSignerCert, BYTE*& pbHash, DWORD& cbHash);

Входные параметры:

fileName – имя файла, содержащее исходное подписанное сообщение
dataFileName – имя файла, предназначенного для хранения данных
pbSignerCert – сертификат ключа подписи в виде набора байт
cbSignerCert – размер сертификата

Выходные параметры:

pbHash – хеш-значение от подписываемых данных
cbHash – размер хеш-значения

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.3.15 Функция AddSignByHashJoinComponents()

Заголовок объявления функции:

BOOL __declspec(dllimport) AddSignByHashJoinComponents(BYTE* pbNewSign, DWORD cbNewSign, LPWSTR outFileName);

Входные параметры:

pbNewSign – отделенное подписанное сообщение (подпись, полученная подписанием хеш-значения)
cbNewSign – размер сообщения
outFileName – имя результирующего файла

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.3.16 Класс AddSignByHash

Переменные класса:

LPWSTR fileName – имя файла, содержащее исходное подписанное сообщение
LPWSTR *fileNameData* – имя файла, содержащее исходные данные
PCCERT_CONTEXT *signerCert* – контекст сертификата
std::vector<BYTE> *hashData* – хеш-значение от данных
std::vector<BYTE> *signatureData* – отделенная подпись от исходного документа
ALG_ID *algid* – алгоритм хеширования, определяемый по сертификату ключа подписи

Внутренние функции:

bool FindExisitingDigestOid() – определение присутствия необходимого поддерживаемого алгоритма хеширования в сообщении; добавление алгоритма в set в случае необходимости
void FreeResourse() – освобождение занимаемых ресурсов

Открытые функции:

1) Конструктор

AddSignByHash(LPWSTR inFileName, LPWSTR dataFileName, BYTE* pbSignerCert, DWORD cbSignerCert)

```
:fileName(inFileName),
fileNameData(dataFileName),
signerCert(nullptr)
{
    signerCert = CertCreateCertificateContext(TYPE_DER, pbSignerCert, cbSignerCert);
}
```

2) Деструктор

```
~AddSignByHash()
{
    FreeResourse();
}
```

3) Функция отделения данных и исходной подписи/подписей, а так же хеширования данных

```
bool SeparateDataAndSing();
```

4) Функция добавления новой подписи к уже существующим в сообщении

```
bool AddNewSignValue(BYTE* pbNewSign, DWORD cbNewSign);
```

5) Функция присоединения данных к подписи

```
bool ConvertContext(LPWSTR outFileName);
```

6) Возврат сформированного хеш-значения

```
std::vector<BYTE> GetHashResult();
```

2.4 Проверка ЭП

2.4.1 Функция GetEncodedFileType()

Заголовок объявления функции:

```
int __declspec(dllimport) GetEncodedFileType(LPWSTR fileName);
```

Входные параметры:

fileName – имя файла сообщения

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение в виде числа, обозначающего тип сообщения (см. таблица 2.3).

Таблица 2.3

Числовое обозначение	Тип сообщения
2	Подписанный
3	Зашифрованный
4	Подписанный и зашифрованный

Если функция завершается ошибкой, возвращаемое значение равно 0.

Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.4.2 Функция IsSignDetached()

Заголовок объявления функции:

BOOL __declspec(dllimport) IsSignDetached(BYTE* context, DWORD contextLength, BOOL *detached);

Входные параметры:

context – блок данных для проверки типа подписи
contextLength – размер блока данных

Выходные параметры:

detached – флаг отделенной подписи

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.4.3 Функция IsSignDetachedByFileName()

Заголовок объявления функции:

BOOL __declspec(dllimport) IsSignDetachedByFileName(LPWSTR fileName, BOOL *detached);

Входные параметры:

filename – имя подписанного файла

Выходные параметры:

detached – флаг отделенной подписи

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.4.4 Функция InitVerifyEds()

Заголовок объявления функции:

BOOL __declspec(dllimport) InitVerifyEds(BOOL getData, LPWSTR outFileName, BOOL detachedEds);

Входные параметры:

getData – флаг формирования данных в результате операции
outFileName – имя результирующего файла (файл с исходными данными)
detachedEds – флаг отделенной подписи

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
if (!BHImportFunctions.InitVerifyEds(getData, outputFileName, false))
{
    var edsException = new EDSException();
    edsException.SetBHError();
    throw (edsException);
}
```

2.4.5 *Функция CryptMsgVerifyUpdate()*

Заголовок объявления функции:

BOOL __declspec(dllimport) CryptMsgVerifyUpdate(BYTE* pbData, DWORD cbData, BOOL lastCall);

Входные параметры:

pbData – блок данных в виде набора байт
cbData – размер блока данных
lastCall – флаг конца данных

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var LastCall = 0;
using (var inputStream = new System.IO.FileStream(inFileName, System.IO.FileMode.Open,
System.IO.FileAccess.Read, System.IO.FileShare.ReadWrite))
{
        while ((bytesRead = inputStream.Read(buffer, 0, SizeBuff)) > 0)
        {
                if (bytesRead < SizeBuff)
                {LastCall = 1;
                }
               if (!BHImportFunctions.CryptMsgVerifyUpdate(buffer, (uint)bytesRead, LastCall))
               {var edsException = new EDSException();
                        edsException.SetBHError();
                        throw (edsException);}
        }
        if (LastCall == 0)
        {LastCall = 1;
                if (!BHImportFunctions.CryptMsgVerifyUpdate(buffer, (uint)0, LastCall))
                {var edsException = new EDSException();
                        edsException.SetBHError();
                        throw (edsException);}
        }
}
```

2.4.6 *Функция VerifyEdsSignature()*

Заголовок объявления функции:

BOOL __declspec(dllimport) VerifyEdsSignature(DWORD& signatureCount);

Выходные параметры:

signatureCount – количество подписей в сообщении

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
uint signatureCount = 0;
if (!BHImportFunctions.VerifyEdsSignature(out signatureCount))
{var edsException = new EDSException();
        edsException.SetBHError();
        throw (edsException);}
```

2.4.7 Функция GetVerifyEdsResult()

Заголовок объявления функции:

void __declspec(dllimport) GetVerifyEdsResult(DWORD signatureIndex, SignInfo& signInfo);

Входные параметры:

signatureIndex – индекс подписи в сообщении

Выходные параметры:

signInfo – структура, содержащая результат проверки подписи

Пример вызова на языке C#:

```
var signatureInfo = new SignatureInfo[signatureCount];
for (int i = 0; i < signatureCount; i++)
{
    SignInfo signInfo;
    BHImportFunctions.GetVerifyEdsResult((uint)i, out signInfo);
    signatureInfo[i] = new SignatureInfo(signInfo);
}
```

2.4.8 Функция VerifyEdsFreeResource()

Заголовок объявления функции:

void __declspec(dllimport) VerifyEdsFreeResource();

Пример вызова на языке C#:

```
BHImportFunctions.VerifyEdsFreeResource();
```

2.4.9 Класс VerifyEDS: public BHGiSCryptography

Переменные класса:

std::vector<OperationResult*> *arrayOfResult* – результаты проверок подписей в сообщении

HCERTSTORE *hCertMemoryStore* – временное хранилище сертификатов из сертификатов, вложенных в подпись

Внутренние функции:

bool FindSignerCert(PCCERT_CONTEXT& signerCert, DWORD signatureIndex) – поиск сертификата подписчика

bool GetAttributes(DWORD signatureIndex, DWORD param, PCRYPT_ATTRIBUTES& attr, DWORD *attr_len) – получение атрибутов подписи (подписанные или неподписанные в зависимости от PCRYPT_ATTRIBUTES attr)

bool VerifyEdsSignatureByCert(DWORD signatureIndex, PCCERT_CONTEXT cert) – проверка математической целостности подписи (по индексу signatureIndex)

bool VerifyEdsSignatureByCertAndHashSet(DWORD signatureIndex, PCCERT_CONTEXT cert, DVCSMessageInfoMessageImprint_t setMessageImprint) – проверка математической целостности подписи(по индексу signatureIndex)

void CreateVerifyResult(BOOL isCounterSignature,DWORD signatureIndex, BOOL result, BYTE* rawCert, DWORD sizeOfRawCert, CMSG_SIGNER_INFO *signerInfo = NULL) – формирование результата проверки ЭП

void CheckMessageForCounterSignature(int signerIndex, PCRYPT_ATTRIBUTES attrib) – проверка подписи на наличие атрибута - заверяющая подпись

bool VerifyCounterSignEds(int signerIndex, BYTE *csData, DWORD csDataSize) – механизм проверки заверяющей подписи

bool VerifyCounterSignEdsByCertInfo(DWORD signerIndex, BYTE* csData, DWORD csDataSize, CERT_INFO certInfo) – проверка математической корректности заверяющей подписи

void FreeResorse() – очистка занятых ресурсов

Открытые функции:

1) Коструктор
VerifyEDS()
:arrayOfResult(0),
hCertMemoryStore(nullptr)
{}
2) Деструктор
~VerifyEDS()
{
FreeResorse();
}
3) Инициализация криптографических параметров проверки подписи
bool InitVerifyEdsParam(BOOL getData, LPWSTR outFileName, BOOL detachedEds);
4) Проверка математической корректности подписи, формирование результата проверки, возвращение количества подписей в сообщении
bool VerifySignature(DWORD& signatureCount);
5) Проверка математической корректности подписи, используя набор хеш-значений от данных для различных алгоритмов хеширования; формирование результата проверки, возвращение количества подписей в сообщении
bool VerifyDetachedSignatureByHashSet(DVCSMessageInfoMessageImprint_t setMessageImprint, DWORD& signatureCount);
6) Проверка корректности штампа времени с доказательствами (усовершенствованный штамп)
bool VerifyTspAdvansedCert(FILETIME tspTime);
7) Возврат результата проверки для подписи
OperationResult* GetResult(DWORD signatureIndex);

2.4.10 Класс VerifyAdvanced

Переменные класса:

AdvancedAttribStruct_t *advancedAttrib* – атрибуты для проверки
std::vector<BYTE> *rawSignature* – значение подписи
std::vector<BYTE> *rawCert* – часть сертификата подписчика
std::vector<CertificateList_t*> *certificateLists* – используемые CRL
std::vector<BasicOCSPResponse_t*> *ocspResponses* – используемые ocsp ответы
std::vector<PCCERT_CONTEXT> *usingCertValues* – значения сертификатов, включенных в подпись
HCERTSTORE *hCertMemoryStore* – дополнительное хранилище сертификатов для построения цепочки
PCCERT_CONTEXT *pSignerCert* – контекст сертификата подписчика
bool *isCorrectProof* - флаг корректности проверки
std::vector<PCCERT_CONTEXT> *ocspCerts* – сертификаты ocsp операторов, используемые при формировании подписи
FILETIME *tspTime* – время создания внешнего штампа времени

Внутренние функции:

bool CheckProofSignerCert() – проверка доказательств сертификата подписчика
bool CheckProofTspCert(PCCERT_CONTEXT pTsaCert, DWORD startCert) – проверка доказательств сертификата tsp оператора
bool CheckProofOcspCerts() – проверка доказательств сертификатов ocsp
bool CheckIncludedCertsInCertValues(PCCERT_CHAIN_CONTEXT ChainContext) – проверка наличия сертификатов из цепочки в атрибуте CertificateValues
bool CheckCertificatesRefs(PCCERT_CHAIN_CONTEXT ChainContext, DWORD startCert) – проверка наличия ссылок на значения сертификатов

bool FindRevValueForCertChain(PCCERT_CHAIN_CONTEXT ChainContext, DWORD startCert) – поиск значений отзывов для цепочки сертификатов

bool CheckRevocationValues() – проверка действительности значений отзывов относительно времени внутреннего штампа

bool CheckRevocationRefs() – проверка наличия ссылок на значения отзывов

bool CheckSignatureTimeStamp() – проверка корректности внутреннего штампа времени (полная проверка всех требований для внутреннего штампа усовершенствованной ЭП)

bool CheckCadesTimeStamp() – проверка корректности внешнего штампа времени и соответствия хеш-значений

PCCERT_CONTEXT FindOcspOperatorCert(BasicOCSPResponse_t* bresponse) – поиск сертификата ocsp оператора

bool CheckTSPItSelf(PCCERT_CONTEXT tspCert, TimeStampToken_t* tspToken, FILETIME tspTime) – проверка действительности доказательств tsp сертификата, используя атрибуты tsp ответа

std::vector<BYTE> VerifyAdvanced::GetDataToExternalTSPRequest() – формирование данных для проверки внешнего штампа времени

void FreeResourse() – очистка памяти

Открытые функции:

1) Конструктор

```
VerifyAdvanced(const AdvancedAttribStruct& advancedAttr, std::vector<BYTE> rawSignature, std::vector<BYTE> rawCert)
        :advancedAttrib(advancedAttr),
        rawSignature(rawSignature),
        rawCert(rawCert),
        certificateLists(0),
        ocspResponses(0),
        usingCertValues(0),
        hCertMemoryStore(nullptr),
        pSignerCert(nullptr),
        isCorrectProof(true),
        ocspCerts(0)
{}
```

2) Конструктор

```
VerifyAdvanced(const AdvancedAttribStruct& advancedAttr, std::vector<BYTE> rawSignature, std::vector<BYTE> rawCert, FILETIME tspTime)
        :advancedAttrib(advancedAttr),
        rawSignature(rawSignature),
        rawCert(rawCert),
        certificateLists(0),
        ocspResponses(0),
        usingCertValues(0),
        hCertMemoryStore(nullptr),
        pSignerCert(nullptr),
        isCorrectProof(true),
        ocspCerts(0),
        tspTime(tspTime)
{}
```

3) Деструктор

```
/// </summary>
~VerifyAdvanced()
{
        FreeResourse();
}
```

4) Проверка доказательств УЭП
bool AdvancedCheckProofs(FILETIME& tspOutFileTime);
5) Проверка доказательств УЭП для штампа времени
bool AdvancedCheckProofsForTsp();
6) Проверка корректности сертификата подписчика по ссылке (ссылка – подписанный атрибут)
static bool CheckSignerCertByCertRefs(PCCERT_CONTEXT verifySignerCert, SigningCertificateV2* signingCertificateV2, SigningCertificate* signingCertificate);

2.4.11 Класс OperationResult

Структура результата проверки ЭП:

```
typedef struct SignInfo
{
	DWORD signatureIndex;
	BOOL isCounterSignature;
	BOOL verifyResult;
	BOOL isAdvanced;
	BOOL verifyCertificateResult;
	BOOL isTimeStampIncluded;
	LPWSTR comment;
	FILETIME signatureTime;
	BYTE* rawSignerCert;
	DWORD rawSignerCertSize;
	BYTE* rawSignatureValue;
	DWORD rawSignatureValueSize;
public:
	SignInfo()
		:signatureIndex(0),
		isCounterSignature(false),
		verifyResult(false),
		isAdvanced(false),
		verifyCertificateResult(false),
		isTimeStampIncluded(false),
		comment(nullptr),
		rawSignerCert(nullptr),
		rawSignerCertSize(0),
		rawSignatureValue(nullptr),
		rawSignatureValueSize(0)
	{
		signatureTime.dwHighDateTime = 0;
		signatureTime.dwLowDateTime = 0;
	}
} SignInfo_t;
```

Переменные класса:

bool result – математическая корректность ЭП
bool countersignature – флаг заверяющей подписи
bool verifyCertificateResult – корректность сертификата
bool isAdvanced – флаг усовершенствованной ЭП
bool isTimeStampIncluded – флаг наличия в подписи штампа времени
DWORD signatureIndex – индекс подписи в CMS
FILETIME filetime – время создания подписи

LPWSTR comment – комментарий к подписи
BYTE *rawCert – сертификат в виде массива байт
DWORD sizeofRawCert – размер массива сертификата
BYTE *signature – значение подписи
DWORD sizeofSignature – длина значения подписи
AdvancedAttribStruct_t* advancedAttrib – атрибуты подписи

Внутренние функции:

bool GetSigningTime (PCRYPT_ATTRIBUTES authAttrib) – получение времени подписи из подписанного атрибута

bool GetComment (PCRYPT_ATTRIBUTES authAttrib) – получение комментария к подписи из подписанного атрибута

bool GetSignCertRefsAttributes(PCRYPT_ATTRIBUTES authAttrib) – получение атрибута SignCertRefsV2

bool GetUnauthenticatedAttributes (PCRYPT_ATTRIBUTES unauthAttrib) – формирование неподписанных атрибутов

void DetectIsAdvanced() – определение типа подписи (усовершенствованная или нет)

bool CheckCertValidity(FILETIME& tspTime) – проверка действительности сертификата в случае обычной ЭП (действительность на текущий момент времени)

void FreeResourse() – очистка памяти

Открытые функции:

1) Конструктор

OperationResult(BOOL counterSignature, DWORD signatureIndex, BOOL result, BYTE* rawCertData, DWORD sizeOfRawCertData)

```
        :result(result),
        counterSignature(counterSignature),
        verifyCertificateResult(false),
        isAdvanced(false),
        isTimeStampIncluded(false),
        signatureIndex(signatureIndex),
        filetime(),
        comment(nullptr),
        rawCert(nullptr),
        sizeofRawCert(sizeOfRawCertData),
        signature(nullptr),
        sizeofSignature(0),
        advancedAttrib(nullptr)
{
        rawCert= new BYTE[sizeofRawCert];
        memcpy(rawCert, rawCertData, sizeofRawCert);
        advancedAttrib = new AdvancedAttribStruct_t();
        //Структура, содержащая значения отзывов сертификатов
        advancedAttrib->revValues=0;
        //Структура, содержащая ссылки на значения отзывов сертификатов
        advancedAttrib->completeRevocationRefs=0;
        //Структура, содержащая значения сертификатов
        advancedAttrib->certificateValues = 0;
        //Структура, содержащая ссылки на значения сертификатов
        advancedAttrib->completeCertificateRefs=0;
        //Структура, содержащая внутренний штамп времени
        advancedAttrib->signatureTimeStamp=0;
        //Структура, содержащая внешний штамп времени
        advancedAttrib->cadesTimeStamp=0;
```

```
		//Структура, содержащая ссылку на значение сертификата подписчика
		advancedAttrib->signingCertificateV2=0;
		filetime.dwHighDateTime = 0;
		filetime.dwLowDateTime = 0;
	}
```

2) Деструктор

```
	~OperationResult()
	{
		FreeResourse();
	}
```

3) Формирование результата проверки подписи

```
	void GenerateResult(PCRYPT_ATTRIBUTES authAttrib, PCRYPT_ATTRIBUTES unauthAttrib, std::vector<BYTE> signBlob);
```

4) Проверка действительности сертификата подписи

```
	void VerifyCertificate();
```

5) Проверка "усовершенствованного" штампа времени

```
	bool VerifyTspAdvansedCertificate(FILETIME tspTime);
```

6) Возврат результата проверки подписи (см. Структуру результата проверки ЭП в данном разделе)

```
	void SetSignInfo(SignInfo& signInfo);
```

7) Возврат результата проверки математической корректности ЭП

```
	bool GetResult()
	{
		return result;
	}
```

8) Возврат результата проверки действительности сертификата

```
	bool GetVerifyCertificateResult()
	{
		return verifyCertificateResult;
	}
```

9) Возврат сертификата подписчика

```
	std::vector<BYTE> GetSignerCert()
	{
		std::vector<BYTE> cert;
		cert.reserve(sizeofRawCert);
		for(int i = 0; i < sizeofRawCert; i++)
			cert.push_back(rawCert[i]);
		return cert;
	}
```

10) Возврат атрибутов УЭП

```
	AdvancedAttribStruct_t* GetAdvancedAttrib()
	{
		return advancedAttrib;
	}
```

2.5 Шифрование

2.5.1 Функция AddReceiverCert()

Заголовок объявления функции:

BOOL __declspec(dllimport) AddReceiverCert(BYTE* receiverCert, DWORD receiverCertLength);

Входные параметры:

receiverCert – сертификат в виде набора байт

receiverCertLength – размер сертификата

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).

Если функция завершается ошибкой, возвращаемое значение равно нулю (false).

Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
foreach (var reseiverCert in Certificates)
{
	if (!BHImportFunctions.AddReceiverCert(reseiverCert.RawData,
	(uint)reseiverCert.RawData.Length))
	{
		var edsException = new EDSException();
		edsException.SetBHError();
		throw (edsException);
	}
}
```

2.5.2 Функция InitEncodeMessage()

Заголовок объявления функции:

BOOL __declspec(dllimport) InitEncodeMessage(LPWSTR outFileName);

Входные параметры:

outFileName – имя результирующего файла

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).

Если функция завершается ошибкой, возвращаемое значение равно нулю (false).

Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
if (!BHImportFunctions.InitEncodeMessage(OutputFileName))
{
	var edsException = new EDSException();
	edsException.SetBHError();
	throw (edsException);
}
```

2.5.3 Функция EncodeMessageUpdate()

Заголовок объявления функции:

BOOL __declspec(dllimport) EncodeMessageUpdate(BYTE* pbData, DWORD cbData, BOOL lastCall);

Входные параметры:

pbData – блок данных в виде набора байт

cbData – размер блока данных

lastCall – флаг конца данных

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var LastCall = 0;
var buffer = new byte[SizeBuff];
var bytesRead = 0;
using (var inputStream = new System.IO.FileStream(inFileName, System.IO.FileMode.Open,
System.IO.FileAccess.Read, System.IO.FileShare.ReadWrite))
{
        while ((bytesRead = inputStream.Read(buffer, 0, SizeBuff)) > 0)
        {if (bytesRead < SizeBuff)
                {
                        LastCall = 1;}
                if (!BHImportFunctions.EncodeMessageUpdate(buffer, (uint)bytesRead, LastCall))
                {
                        var edsException = new EDSException();
                        edsException.SetBHError();
                        throw (edsException);}
        }
         if (LastCall == 0)
        {
                LastCall = 1;
                if (!BHImportFunctions.EncodeMessageUpdate(buffer, (uint)0, LastCall))
                {
                         var edsException = new EDSException();
                        edsException.SetBHError();
                         throw (edsException);}
        }
}
```

2.5.4 Класс EncryptClass: public BHGiSCryptography

Переменные класса:

std::vector<PCCERT_CONTEXT> arrayCerts – массив сертификатов получателей
CMSG_ENVELOPED_ENCODE_INFO envelopedEncodeInfo – структура, содержащая информацию для шифрования данных

Открытые функции:

1) Конструктор

```
EncryptClass()
{}
```

2) Деструктор

```
~EncryptClass()
{
        for(int i =0; i < arrayCerts.size(); i++)
                CertFreeCertificateContext(arrayCerts.at(i));
        arrayCerts.clear();
};
```

3) Добавление сертификата в список получателей

```
bool AddReceiverCert(BYTE* receiverCert, DWORD receiverCertLength);
```

4) Установка параметров шифрования

```
bool SetEncryptionParam(LPWSTR outFileName);
```

2.6 Расшифрование

2.6.1 Функция InitDecodeMessage()

Заголовок объявления функции:

BOOL __declspec(dllimport) InitDecodeMessage(LPWSTR outFileName);

Выходные параметры:

outFileName – имя результирующего файла

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Привем вызова на языке C#:

```
if (!BHImportFunctions.InitDecodeMessage(OutputFileName))
{var edsException = new EDSException();
        edsException.SetBHError();
        throw (edsException);
}
```

2.6.2 Функция DecodeMessageUpdate()

Заголовок объявления функции:

BOOL __declspec(dllimport) DecodeMessageUpdate(BYTE* pbData, DWORD cbData, BOOL lastCall);

Входные параметры:

pbData – блок данных в виде набора байт
cbData – размер блока данных
lastCall – флаг конца данных

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
using (var inputStream = new System.IO.FileStream(inFileName, System.IO.FileMode.Open,
System.IO.FileAccess.Read, System.IO.FileShare.ReadWrite))
{while ((bytesRead = inputStream.Read(buffer, 0, SizeBuff)) > 0)
        {
                if (bytesRead < SizeBuff)
                {LastCall = 1;}
               if (!BHImportFunctions.DecodeMessageUpdate(buffer, (uint)bytesRead, LastCall))
               {var edsException = new EDSException();
                        edsException.SetBHError();
                        throw (edsException);}
        }
         if (LastCall == 0)
        {LastCall = 1;
                if (!BHImportFunctions.DecodeMessageUpdate(buffer, (uint)0, LastCall))
                {
                        var edsException = new EDSException();
                        edsException.SetBHError();
                        throw (edsException);}
        }
}
```

2.6.3 Функция GetDecodedCert()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetDecodedCert(BYTE*& pbDecodedCert, DWORD& cbDecodedCert)

Выходные параметры:

pbDecodedCert – сертификат в виде набора байт

cbDecodedCert – размер сертификата

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).

Если функция завершается ошибкой, возвращаемое значение равно нулю (false).

Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

Пример вызова на языке C#:

```
var certPtr = IntPtr.Zero;
uint certSize;
if (!BHImportFunctions.GetDecodedCert(ref certPtr, out certSize))
{
    var edsException = new EDSException();
    edsException.SetBHError();
    throw (edsException);
}
```

2.6.4 Класс DecryptClass: public BHGiSCryptography

Переменные класса:

bool isSecretKeyFind – флаг определения закрытого ключа

PCCERT_CONTEXT userCert – сертификат получателя (сертификат с привязкой к закрытому ключу для расшифровывания)

Открытые функции:

1) Конструктор

```
DecryptClass()
    :isSecretKeyFind(false),
    userCert(nullptr)
{}
```

2) Деструктор

```
~DecryptClass()
{
    if(userCert)
        CertFreeCertificateContext(userCert);
}
```

3) Установка параметров расшифровывания

```
bool SetDecryptionParam(LPWSTR outFileName);
```

4) Возврат сертификата расшифровывания

```
PCCERT_CONTEXT GetDecryptionCert();
```

5) Возврат флага закрытого ключа

```
bool IsSecretKey()
{
    return isSecretKeyFind;
}
```

6) Определение подходящего закрытого ключа для расшифровывания

```
bool FindSecretKey();
```

2.7 DVCS-сервис

2.7.1 Функция SignDVCSData()

Заголовок объявления функции:

BOOL __declspec(dllimport) SignDVCSData(BYTE* pbDVCSData, DWORD cbDVCSData, BYTE* pbSignerCert, DWORD cbSignerCert, LPSTR ContentObjID, LPSTR pinCode, BYTE* &pbSignedDVCSData, DWORD& cbSignedDVCSData);

Входные параметры:

pbDVCSData – данные для подписи
cbDVCSData – размер данных для подписи
pbSignerCert – сертификат подписчика в виде набора байт
cbSignerCert – размер сертификата подписчика
ContentObjID – объектный идентификатор типа данных
pinCode – пин-код для доступа к контейнеру закрытого ключа

Выходные параметры:

pbSignedDVCSData – подпись в виде набора байт
cbSignedDVCSData – размер сформированной подписи

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.2 Функция VerifyDVCSSignature()

Заголовок объявления функции:

BOOL __declspec(dllimport) VerifyDVCSSignature(BYTE* pbSignedDVCSData, DWORD cbSignedDVCSData, BYTE* &pbDVCSData, DWORD& cbDVCSData, SignInfo& signInfo);

Входные параметры:

pbSignedDVCSData – подпись для проверки
cbSignedDVCSData – размер подписи

Выходные параметры:

pbDVCSData – данные, полученные при снятии подписи
cbSignerCert – размер данных
signInfo – результат проверки подписи

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.3 Функция CreateDVCSRequest()

Заголовок объявления функции:

BOOL __declspec(dllimport) CreateDVCSRequest(e_ServiceType serviceType, BYTE* pbData, DWORD cbData, LPSTR hashAlg,BYTE* &pbDVCSRequest, DWORD& cbDVCSRequest);

Входные параметры:

serviceType – тип запроса
pbData – данные для запроса в виде набора байт
cbData – размер данных для запроса
hashAlg – алгоритм хеширования

Выходные параметры:

pbDVCSRequest – сформированный запрос в виде набора байт
cbDVCSRequest – размер запроса

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.4 Функция CreateHashVsdRequest()

Заголовок объявления функции:

BOOL __declspec(dllimport) CreateHashVsdRequest(LPWSTR InFileName, BYTE* &pbDVCSRequest, DWORD& cbDVCSRequest)

Входные параметры:

InFileName – имя исходного проверяемого подписанного файла

Выходные параметры:

pbDVCSRequest – сформированный запрос в виде набора байт
cbDVCSRequest – размер запроса

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.5 Класс DVCRequest

Переменные класса:

e_ServiceType *serviceType* – тип запроса (значения переменной см. в таблице 2.4).
Таблица 2.4

Тип запроса	Значение переменной
cpd	1
vsd	2
vpkc	3
ccpd	4

BYTE* *pbData* – блок данных для запроса
DWORD *cbData* – размер данных
LPSTR *hashAlg* – аглоритм хеширования (в случае ccpd запроса)
DVCSRequest_t* *dvcsRequest* – структура, содержащая DVCS запрос
LPWSTR *InFileName* – имя файла для проверки
BOOL *usingHash* – флаг использования хеширования данных

Внутренние функции:

bool CreateDVCSData() – заполнение тела запроса
bool CreateVsdHashData() – заполнение тела vsd запроса хеш-запросом
bool CreateDVCSRequestInformation() – заполнение запроса информацией
void FreeResourse() – освобождение занятых ресурсов

Открытые функции:

1) Конструктор

```
DVCRequest(e_ServiceType serviceType, BYTE* pbData, DWORD cbData, LPSTR hashAlg)
        :serviceType(serviceType),
        pbData(pbData),
        cbData(cbData),
        hashAlg(hashAlg),
        dvcsRequest(0),
        InFileName(nullptr),
        usingHash(false)
{dvcsRequest=(DVCSRequest_t*)calloc(1,sizeof *dvcsRequest);
}
```

2) Конструктор

```
DVCRequest(LPWSTR InFileName)
        :serviceType(ServiceType_vsd),
        pbData(nullptr),
        cbData(0),
        hashAlg(nullptr),
        dvcsRequest(0),
        InFileName(InFileName),
        usingHash(true)
{
        dvcsRequest=(DVCSRequest_t*)calloc(1,sizeof *dvcsRequest);
}
```

3) Деструктор

```
~DVCRequest()
{
        FreeResourse();
}
```

4) Старт операции создания dvc запроса

```
bool CreateDVCRequest();
```

5) Получение результата

```
std::vector<BYTE> GetDVCRequestData();
```

2.7.6 Функция GetOIDsFromDVCSRequest()

Заголовок объявления функции:

```
BOOL __declspec(dllimport) GetOIDsFromDVCSRequest(BYTE* pbDVCSRequest, DWORD cbDVCSRequest, DWORD& countOIDs, wchar_t** &OIDs);
```

Входные параметры:

pbDVCSRequest – DVCS-запрос в виде набора байт

cbDVCSRequest – размер DVCS-запроса

Выходные параметры:

countOIDs – количество алгоритмов

OIDs – список алгоритмов

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).

Если функция завершается ошибкой, возвращаемое значение равно нулю (false).

Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.7 Функция CreateDVCSResponse()

Заголовок объявления функции:

BOOL __declspec(dllimport) CreateDVCSResponse(BYTE* pbDVCSRequest, DWORD cbDVCSRequest, BYTE* pbDVCSCert, DWORD cbDVCSCert, DWORD dvcSerialNumber, LPWSTR tspAdress, BYTE* &pbDVCSResponse, DWORD& cbDVCSResponse);

Входные параметры:

pbDVCSRequest – DVCS-запрос в виде набора байт
cbDVCSRequest – размер DVCS-запроса
pbDVCSCert – сертификат службы DVCS в виде набора байт
cbDVCSCert – размер сертификата
dvcSerialNumber – серийный номер создаваемой квитанции
tspAdress – адрес используемой службы штампов времени

Выходные параметры:

pbDVCSResponse – сформированный DVCS-ответ в виде массива байт
cbDVCSResponse – размер DVCS-ответа

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.8 Функция CreateDVCSErrorNotice()

Заголовок объявления функции:

BOOL __declspec(dllimport) CreateDVCSErrorNotice(int pkiFailureInfo, LPWSTR pkiFreeText, BYTE* &pbDVCSErrorNotice, DWORD& cbDVCSErrorNotice);

Входные параметры:

pkiFailureInfo – тип ошибки
pkiFreeText – описание ошибки

Выходные параметры:

pbDVCSErrorNotice – сформированный DVCS-ответ в виде массива байт
cbDVCSErrorNotice – размер DVCS-ответа

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.9 Функция ChangeDVCSerial()

Заголовок объявления функции:

BOOL __declspec(dllimport) ChangeDVCSerial(BYTE* pbDVCSResponse, DWORD cbDVCSResponse, DWORD newSerialNumber, BYTE* &pbUpdateDVCSResponse, DWORD& cbUpdateDVCSResponse);

Входные параметры:

pbDVCSResponse – исходный DVCS-ответ
cbDVCSResponse – размер исходного DVCS-ответа
newSerialNumber – новый серийный номер

Выходные параметры:

pbUpdateDVCSResponse – обновленный DVCS-ответ в виде массива байт
cbUpdateDVCSResponse – размер обновленного DVCS-ответа

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.10 Класс DVCResponse

Таблица 2.5. Константы

Константа (тип LPWSTR)	**Содержание**
ParsingError	Ошибка декодирования запроса
VerifySigatureFatalError	Не поддерживаемый формат электронного документа
ServerInitCertError	Внутренняя ошибка сервера. Ошибка инициализации сертификата
ServerHashDataError	Внутренняя ошибка сервера. Ошибка хеширования данных
IncorrectSignature	Подпись математически некорректна
InvalidCertificate	Сертификат подписчика недействителен
CertCRLRevoked	Сертификат отозван по CRL
CertOCSPRevoked	Сертификат отозван по OCSP
RevokedInfoInaccessible	Информация об отзыве сертификата недоступна
CertFormatError	Неверный формат проверяемого сертификата

Структура – результат проверки сертификата:

```
typedef struct CertVerifyInfo
{
        PKIStatusInfo_t pkiStatusInfo;
        PCCERT_CHAIN_CONTEXT certChainContext;
        OCSPResponse_t* ocspresponse;
        CertificateList_t* certificateList;
}CertVerifyInfo_t;
```

Переменные класса:

BYTE* *pbDVCSRequest* – DVCS-запрос
DWORD *cbDVCSRequest* – размер запроса
BYTE* *pbDVCSCert* – сертификат службы DVCS
DWORD *cbDVCSCert* – размер сертификата
long *dvcSerialNumber* – серийный номер квитанции
LPWSTR *tspAdress* – адрес службы штампов времени
DVCSResponse_t* *dvcsResponse* – структура DVC-квитанции
DVCSCertInfo_t* *dvcsCertInfo* – структура успешного результата проверки
DVCSRequest_t* *dvcsRequest* – структура DVCS-запроса
DWORD *pkiFailureInfo* – int для ошибки
LPWSTR *errorNoticeString* – строка для ошибки
std::vector<BYTE> *hash* – хеш-значение
ALG_ID *hashAlg* – алгоритм хеширования, используемый для операции
HCRYPTPROV *hProv* – контекст криптопровайдера

Внутренние функции:

DVCSTime_t CreateDVCSTime() – формирование времени создания DVCS-ответа
bool CreateCPDResponse() – формирование ответа на запрос типа CPD
bool CreateVSDResponse() – формирование ответа на запрос типа VSD
bool CreateCPKCResponse() – формирование ответа на запрос типа CPKC
bool CreateCCPDResponse() – формирование ответа на запрос типа CCPD
bool VerifyVsdData(VerifyEDS*& verifyEDS, DWORD& signatureCount) – проверка подписи в случае запроса типа VSD
void VerifyCpkcData(CertVerifyInfo_t*& certVerifyInfo, PCCERT_CONTEXT certToVerify) – проверка подписи в случае запроса типа CPKC
void FreeResourse() – очистка ресурсов

Открытые функции:

1) Конструктор

```
DVCResponse(BYTE* pbDVCSRequest, DWORD cbDVCSRequest, long dvcSerialNumber, BYTE* pbDVCSCert, DWORD cbDVCSCert, LPWSTR tspAdress)
        :pbDVCSRequest(pbDVCSRequest),
        cbDVCSRequest(cbDVCSRequest),
        pbDVCSCert(pbDVCSCert),
        cbDVCSCert(cbDVCSCert),
        dvcSerialNumber(dvcSerialNumber),
        tspAdress(tspAdress),
        dvcsResponse(0),
        dvcsCertInfo(0),
        dvcsRequest(0),
        pkiFailureInfo(-1),
        errorNoticeString(nullptr),
        hashAlg(0),
        hProv(0)
{
        dvcsResponse=(DVCSResponse_t*)calloc(1, sizeof *dvcsResponse);
        dvcsCertInfo=(DVCSCertInfo_t*)calloc(1, sizeof *dvcsCertInfo);
}
```

2) Конструктор для DVC-error

```
DVCResponse(DWORD pkiFailureInfo, LPWSTR pkiFreeText)
        :pbDVCSCert(nullptr),
        cbDVCSCert(0),
        dvcsResponse(0),
        dvcsCertInfo(0),
        dvcsRequest(0),
        pkiFailureInfo(pkiFailureInfo),
        errorNoticeString(pkiFreeText),
        hashAlg(0),
        hProv(0)
{
        dvcsResponse=(DVCSResponse_t*)calloc(1, sizeof *dvcsResponse);
}
```

3) Конструктор

```
~DVCResponse()
{
        FreeResourse();
}
```

4) Формирование структуры корректного ответа сервера
bool CreateDVCSResponse();
5) Формирование структуры ошибки
void CreateDVCSErrorNotice();
6) Формирование структуры ошибки с параметрами импортируемой функции
void CreateDVCSErrorNotice(DWORD pkiFailureInfo, LPWSTR pkiFreeText);
7) Возвращаем результат
std::vector<BYTE> GetResponseData();

2.7.11 Функция GetAttrFromSign()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetAttrFromSign(BYTE* pbSignData, DWORD cbSignData, LPWSTR attrType, DWORD signatureIndex, SingAttributes_t &signAttributes);

Входные параметры:

pbSignData – подписанный файл в виде набора байт
cbSignData – размер файла
attrType – OID атрибута
signatureIndex – индекс подписи в сообщении

Выходные параметры:

signAttributes – структура с найденными атрибутами

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.12 Функция AddAttrToSign()

Заголовок объявления функции:

BOOL __declspec(dllimport) AddAttrToSign(BYTE* pbSignData, DWORD cbSignData, DWORD signatureIndex, bh_Attribute_t bh_attribute_t, BYTE* &pbNewSignData, DWORD& cbNewSignData);

Входные параметры:

pbSignData – подписанный файл в виде набора байт
cbSignData – размер файла
signatureIndex – индекс подписи в сообщении
bh_attribute_t – структура, описывающая добавляемый атрибут
pbNewSignData – подписанный файл с вновь добавленным атрибутов в виду набора байт
cbNewSignData – размер сформированного файла

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.13 Функция ParseDVCResponce()

Заголовок объявления функции:

BOOL __declspec(dllimport) ParseDVCResponce(BYTE* pbDVCResponce, DWORD cbDVCResponce, DVCSResponseInfoStruct& dvcResponseInfo);

Входные параметры:

pbDVCResponce – DVCS-ответ в виде набора байт
cbDVCResponce – размер DVCS-ответа
dvcResponseInfo – структура, содержащая информацию из DVCS-ответа

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.7.14 Класс DVCResponceParsing

Структура представления времени в квитанции:

```
typedef struct DVCSTimeStruct
{
	DWORD timePresent;
	FILETIME fileTime;
	DWORD tokenLen;
	BYTE* timeStampToken;
public:
	DVCSTimeStruct()
		:timePresent(0),
		tokenLen(0),
		timeStampToken(nullptr)
	{}
};
```

Структура статуса операции:

```
typedef struct DVCSPKISatusInfoStruct
{
	DWORD size;
	BYTE* pkiStatus;
public:
	DVCSPKISatusInfoStruct()
		:size(0),
		pkiStatus(nullptr)
	{}
};
```

Структура информации о запросе:

```
typedef struct DVCSRequestInfoStruct
{
	DWORD serviceType;
	DVCSTimeStruct* requestTime;
public:
	DVCSRequestInfoStruct()
		:serviceType(0),
		requestTime(nullptr)
	{}
};
```

Структура DVC-квитанции с положительным результатом:

```
typedef struct DVCSOkInfoStruct
{
	DVCSRequestInfoStruct* reqInfo; //Информация о запросе
	DWORD responseSerial; //Серийный номер квитанции
	DVCSTimeStruct* responseTime; //Время формирования квитанции
	DVCSPKISatusInfoStruct* dvStatus; //Статус операции
	DWORD certsCount; //Кол-во элементов результатов
	DVCSTargetEtcChainStruct** certs; //Подробрый результат проверки
```

```
public:
	DVCSOkInfoStruct()
		:reqInfo(nullptr),
		responseSerial(0),
		responseTime(nullptr),
		dvStatus(nullptr),
		certsCount(0),
		certs(nullptr)
	{}
};
```

Структура DVC-квитанции с ошибочным результатом:

```
typedef struct DVCSErrorInfoStruct
{
	DVCSPKISatusInfoStruct* transactionStatus;
public:
	DVCSErrorInfoStruct()
		:transactionStatus(nullptr)
	{}
};
```

Дополнительные компоненты:

```
typedef struct DVCSCertEtcTokenStruct
{
	DWORD type;
	DWORD dataLen;
	BYTE* data;
public:
	DVCSCertEtcTokenStruct()
		:type(0),
		dataLen(0),
		data(nullptr)
	{}
};
```

Структура дополнительной информации о компонентах:

```
typedef struct DVCSTargetEtcChainStruct
{
	DVCSCertEtcTokenStruct* target;
	DWORD chainCount;
	DVCSCertEtcTokenStruct** chain;
public:
	DVCSTargetEtcChainStruct()
		:target(nullptr),
		chainCount(0),
		chain(nullptr)
	{}
};
```

Переменные класса:

std::vector<BYTE> *responseData* – байты квитанции

DVCSResponse_t* *dvcsResponse* – структура DVC-квитанции

DVCSResponseInfoStruct *dvcResponseInfo* – информация о проверке (возвращаемое значение):

```
typedef struct DVCSResponseInfoStruct
{
	DWORD responseStatus;
	DVCSErrorInfoStruct* dvcErrorInfo;
	DVCSOkInfoStruct* dvcOkInfo;
public:
	DVCSResponseInfoStruct()
		:responseStatus(0),
		dvcErrorInfo(nullptr),
		dvcOkInfo(nullptr)
	{}
};
```

void *FreeResourse()* – очистка памяти

Открытые функции:

1) Конструктор

```
DVCResponceParsing(BYTE* pbDVCResponce, DWORD cbDVCResponce)
	:dvcsResponse(0)
{
	responseData.reserve(cbDVCResponce);
	for(int i = 0; i < cbDVCResponce; i++)
		responseData.push_back(pbDVCResponce[i]);
}
```

2) Деструктор

```
~DVCResponceParsing()
{
	FreeResourse();
}
```

3) Запуск операции анализа

```
bool StartParsing();
```

4) Формирование ошибочного результата проверки DVCS-запроса

```
void FormingDVCSErrorInfo();
```

5) Формирование положительного результата проверки DVCS-запроса

```
void FormingDVCSOkInfo();
```

6) Возврат информации о проверке

```
DVCSResponseInfoStruct GetResponseInfo()
{
	return dvcResponseInfo;
}
```

Дополнительные функции:

DVCSRequestInfoStruct* ParseDVCSRequestInfo(DVCSRequestInformation_t* reqInfo) – анализ структуры RequestInfo

DVCSTimeStruct* ParseDVCSTime(DVCSTime_t* dvcsTime) – анализ времени

DVCSPKISatusInfoStruct* ParseDVCSPKIStatus(PKIStatusInfo_t* pkiStatus) – анализ статуса

DVCSTargetEtcChainStruct** ParseDVCSTargetEtcChain(certsDVCSCertInfo_t* certs) – анализ вложений

DVCSCertEtcTokenStruct* ParseDVCSCertEtcToken(DVCSCertEtcToken_t* certToken) – анализ вложений

2.8 Сопутствующие

2.8.1 Функция GetOcspResponseByCert()

Заголовок объявления функции:

BOOL __declspec(dllimport) GetOcspResponseByCert(BYTE* pbCert, DWORD cbCert, BYTE*& pbOcspResponse, DWORD &cbOcspResponse);

Входные параметры:

pbCert – проверяемый сертификат в виде набора байт

cbCert – размер сертификата

Выходные параметры:

pbOcspResponse – OCSP-ответ в виде набора байт

cbOcspResponse – размер OCSP-ответа

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.8.2 Класс OcspManagment

Переменные класса:

PCCERT_CONTEXT *certToCheck* – сертификат для проверки
PCCERT_CONTEXT *ocspCert* – сертификат OCSP оператора
BSTR* *ocspAdress* – массив адресов точек распространения OCSP
DWORD *countAdress* – количество адресов
std::vector<BYTE> *ocspResponseData* – ответ в виде массива байт
OCSPRequest_t* *ocspRequest* – структура OCSP-запроса
OCSPResponse_t* *ocspResponse* – структура OSCP-ответа
BasicOCSPResponse_t* *basicResponse* – структура basicResponse

Внутренние функции:

bool CreateOcspRequest() – формирование OCSP-запроса
bool GetOcspResponse() – получение OCSP-ответа
void FreeResourse() – очистка занятых ресурсов

Открытые функции:

1) Конструктор, используемый при создании подписи

```
OcspManagment(PCCERT_CONTEXT Cert)
        :certToCheck(Cert),
        ocspCert(NULL),
        ocspAdress(NULL),
        countAdress(0),
        ocspRequest(0),
        ocspResponse(0),
        basicResponse(0)
{}
```

2) Конструктор, используемый при проверке подписи

```
OcspManagment(BasicOCSPResponse_t* bResponse)
        :certToCheck(NULL),
        ocspCert(NULL),
        ocspAdress(NULL),
        countAdress(0),
        ocspRequest(0),
        ocspResponse(0),
        basicResponse(0)
```

```
{
        std::vector<BYTE> basicResponseData =
        Common::GetEncode(&asn_DEF_BasicOCSPResponse, bResponse);
        basicResponse =
        (BasicOCSPResponse_t*)Common::GetDecode(&asn_DEF_BasicOCSPResponse,
        basicResponseData);
}
3) Деструктор
~OcspManagment()
{
        FreeResourse();
}
4) Проверка действительности сертификата по OCSP
bool VerifyCertByOcsp();
5) Проверка подписи OCSP-ответа
bool CheckOcspSignature(PCCERT_CONTEXT cert);
6) Получение времени формирования OCSP-ответа
FILETIME GetOcspTime();
7) Получение сертификата OCSP-оператора
PCCERT_CONTEXT GetOcspCert();
8) Получение статуса проверки сертификата
long GetResponseStatus();
9) Возврат структуры BasicOCSPResponse
BasicOCSPResponse_t* GetBasicOCSPResponse();
10) Возврат структуры OCSPResponse
OCSPResponse_t* GetOCSPResponse();
11) Возврат ответа ocsp в виде массива байт
std::vector<BYTE> GetResponseByte();
```

2.8.3 Функция SecureDeleteFile()

Заголовок объявления функции:

```
BOOL __declspec(dllimport) SecureDeleteFile(LPWSTR fileName);
```

Входные параметры:

fileName – имя файла для удаления

Возвращаемое значение:

Если функция завершается успешно, возвращаемое значение отлично от нуля (true).
Если функция завершается ошибкой, возвращаемое значение равно нулю (false).
Для получения информации об ошибке вызовите функцию GetBHErrorInfo().

2.8.4 Класс SecureDeleteClass

Переменные класса:

LPWSTR *filename* – имя файла для гарантированного удаления
bool *CleanCompressedFiles*

Внутренние функции:

```
bool SecureDeleteCompressed();
bool SecureDelete(DWORD SizeHigh, DWORD SizeLow);
```

Открытые функции:

```
1) Конструктор
SecureDeleteClass(LPWSTR fileName)
        :fileName (fileName)
```

```
{
        CleanCompressedFiles = false;
};
2) Деструктор
~SecureDeleteClass()
{};
3) bool LocateNativeEntryPoints();
4) Запуск операции на выполнение
bool ProcessFile();
```

2.8.5 Функция SetProxyNameAndPassword()

Заголовок объявления функции:

```
void __declspec(dllimport) SetProxyNameAndPassword(BSTR name, BSTR pass);
```

Входные параметры:

name – имя пользователя
pass – пароль

Пример вызова на языке C#:

```
BHImportFunctions.SetProxyNameAndPassword(name, pass);
```

2.8.6 Класс HttpDownloader

Переменные класса:

bool *isPost* – флаг типа запроса (post, get)
BSTR *site* – адрес
BSTR *fname* – имя файла
std::vector<BYTE> *request* – запрос в виде массива байт
HINTERNET *hRequest* – заголовок соединения

Внутренние функции:

bool SendHttpRequest() – отправка запроса

Открытые функции:

```
1) Конструктор
HttpDownloader(bool isPost, BSTR site, BSTR fname, std::vector<BYTE> request)
        :isPost(isPost),
        site(site),
        fname(fname),
        request(request),
        hRequest(NULL)
{};
2) Деструктор
~HttpDownloader(void)
{};
3) Формирование и возврат полученного ответа
bool GetHttpResponse(std::vector<BYTE> &response);
```

2.8.7 Класс ParseUrl

Открытые функции:

```
1) Анализ адреса
static void ParseUrlString(BSTR Adress, BSTR* site, BSTR* fname);
```

2.8.8 Класс TspManagment

Переменные класса:

BSTR *tspAdress* – адрес службы штампов времени
DWORD *algorithmOid* – алгоритм хеширования
std::vector<BYTE> *hash* – хеш для tsp-запроса
TimeStampReq_t* *tsprequest* – структура tsp-запроса
TimeStampResp_t* *tspresponse* – структура tsp-ответа
TimeStampToken_t **token* – структура tsp token – сам неподписанный ответ
PCCERT_CONTEXT *pTspCert* – сертификат tsp оператора
bool *isOldVersion* – версия tsp (флаг необходим для проверки нового формата штампа времени от КриптоПро)

Внутренние функции:

void CreateTspRequest() – создание tps запроса
bool GetTspResponse() – получение tsp ответа
std::vector<BYTE> GetEncodedStampToken() – кодировка структуры TimeStampToken_t *token
std::vector<BYTE> GetEncodedContent() – формирование ответа сервера в чистом виде
bool FindTSOperatorCert() – поиск сертификата tsp оператора в структуре tsp-ответа
void FreeResourse() – очистка занятых ресурсов

Открытые функции:

1) Конструктор, используемый при создании подписи

```
TspManagment(BSTR adress, DWORD oid, std::vector<BYTE> hashToTimeStamping)
        :tspAdress(adress),
        algorithmOid(oid),
        hash(hashToTimeStamping),
        tsprequest(0),
        tspresponse(0),
        token(0),
        pTspCert(NULL),
        isOldVersion(true)
{}
```

2) Конструктор, используемый при проверке подписи

```
TspManagment(TimeStampToken_t *tsToken)
        :tspAdress(NULL),
        algorithmOid(0),
        hash(0),
        tsprequest(0),
        tspresponse(0),
        token(0),
        pTspCert(NULL),
        isOldVersion(true)
{
            std::vector<BYTE> tokenData =
        Common::GetEncode(&asn_DEF_TimeStampToken, tsToken);
            token = (TimeStampToken_t *)Common::GetDecode(&asn_DEF_TimeStampToken,
        tokenData);
            FindTSOperatorCert();
}
```

3) Деструктор
~TspManagment()
{
FreeResourse();
}
4) Формирование tsp
bool CreateTimeStamp();
5) Проверка подписи на tsp-ответе
bool CheckTspSignature();
6) Проверка соответствия хеш-значений
bool TspCompareHash(std::vector<BYTE> hash);
7) Получение времени создания штампа
FILETIME GetTspTime();
8) Получение сертификата tsp оперетора
PCCERT_CONTEXT GetTspCert();
9) Возврат token
TimeStampToken_t* GetToken();
10) Возврат версии штампа
bool GetTpsVersion();

2.8.9 Класс общих операций BHGiSCryptography

Переменные класса:

HCRYPTMSG hMsg – дескриптор сообщения
HANDLE hOutMsgFile – указатель на выходные данные
PCMSG_STREAM_INFO stStreamInfo – структура для потоковой обработки информации

Открытые функции:

1) Конструктор
BHGiSCryptography()
:hMsg(nullptr),
hOutMsgFile(INVALID_HANDLE_VALUE),
stStreamInfo(nullptr)
{}
2) Деструктор
virtual ~BHGiSCryptography()
{
if(hOutMsgFile != NULL)
{
CloseHandle(hOutMsgFile);
hOutMsgFile = NULL;
}
CryptMsgClose(hMsg);
if(stStreamInfo)
delete stStreamInfo;
};
3) Установка параметров для потоковой обработки данных
bool SetOutputParam(LPWSTR outFileName, bool isNullCallback);
4) Открытие дескриптора на кодирование
bool SethMsgOpenToEncode(DWORD dwFlags, DWORD dwMsgType, void const *pvMsgEncodeInfo, LPSTR pszInnerContentObjID = NULL);

5) Открытие дескриптора на декодирование
bool SethMsgOpenToDecode(DWORD dwFlags);
6) Обновление дескриптора данными
bool CryptMessageUpdate(BYTE* pbData, DWORD cbData, BOOL lastCall);
7) Получение дескриптора сообщения
HCRYPTMSG GetHMsg();
8) cms сообщение
std::vector<BYTE> GetEncodedMessage(DWORD flag);
9) Закрытие выходного файла
void CloseFileHandle();
10) Запись данных в файл
bool WriteDataFile(std::vector<BYTE> data);
11) Получение числового параметра из сообщения
bool GetMsgParam(DWORD paramType, DWORD& data);

2.8.10 Класс общих вспомогательных функций Common

Открытые функции:

1) Формирование хеш-значения от заданных данных
static bool CalculateHash(ALG_ID alg, HCRYPTPROV hProv, BYTE* pbContent, DWORD cbContent, std::vector<BYTE> &hash);
2) Получение подходящего криптопровайдера по заданному алгоритму
static bool GetAppropriateCPByAlgID(ALG_ID calg, HCRYPTPROV &hProvider);
3) Получение OID-а алгоритма определенного типа по типу криптопровайдера
static ALG_ID GetAvailableCryptoprovAlgsByProvType(DWORD provType, DWORD algType);
4) Получение числового значения подписи по индексу из сообщения
static std::vector<BYTE> GetSignOnlyFromHmsg(HCRYPTMSG hMsg, DWORD signedIndex);
5) Получение сертификата из сообщения по индексу (поиск сертификата для конкретной подписи)
static PCCERT_CONTEXT GetSignerCertFromHmsg(HCRYPTMSG hMsg, DWORD signedIndex);
6) Получение сертификата из сообщения (из добавленных в сообщение сертификатов)
static PCCERT_CONTEXT GetCertFromHmsg(HCRYPTMSG hMsg, DWORD certIndex);
7) Кодирование данных структуры asn.1
static std::vector<BYTE> GetEncode(asn_TYPE_descriptor_t *type_descriptor, void *struct_ptr);
8) Декодирование данных в структуру asn.1
static void* GetDecode(asn_TYPE_descriptor_t *type_descriptor, std::vector<BYTE> data);
9) Декодирование данных в структуру asn.1
static void* GetDecode(asn_TYPE_descriptor_t *type_descriptor, BYTE* data, DWORD dataLen);
10) Конвертирование GeneralizedTime в FileTime
static FILETIME ConvertGeneralizedTimeToFileTime(const GeneralizedTime_t *st);
11) Определение алгоритма по информации из структуры
static ALG_ID GetAlgIDByObjectIdentifier(OBJECT_IDENTIFIER_t *oid);
12) Определение OID-а
static LPSTR GetOIDByByObjectIdentifier(OBJECT_IDENTIFIER_t *oid);
13) Создание хранилища сертификатов для проверки на основе сертификатов, вложенных в подпись
static HCERTSTORE CreateMemoryStore(std::vector<PCCERT_CONTEXT> usingCertValues);
14) Конвертирование unicode строки в ansi строку
static LPSTR ConvertWideCharToMultiByte(DWORD flag, LPWSTR utfString);
15) Конвертирование ansi строки в unicode строку
static LPWSTR ConvertMultiByteToWideChar(DWORD flag, LPSTR multiByteString);

16) Установка ошибки, связанной с проверкой цепочки сертификации
static void SetCertChainError(DWORD chainError);
17) Функция-транслятор для перехвата структурных исключений
static void Trans_func(unsigned int u, EXCEPTION_POINTERS* pExp);

2.9 Работа с ошибками

2.9.1 Функция GetErrors()

Заголовок объявления функции:

void __declspec(dllimport) GetErrors(DWORD& LastError, DWORD*& BHGiSErrorCode, DWORD& countBHGiSError);

Выходные параметры:

LastError – код системной ошибки
BHGiSErrorCode – массив кодов ошибок модуля
countBHGiSError – количество кодов ошибок модуля

2.9.2 Функция GetBHErrorInfo()

Заголовок объявления функции:

void __declspec(dllimport) GetBHErrorInfo(BHErrorInfo& bhErrorInfo);

Выходные параметры:

bhErrorInfo – структура, содержащая информацию об ошибке

Пример вызова на языке C#:

```
BHImportFunctions.BHErrorInfo bhErrorInfo;
BHImportFunctions.GetBHErrorInfo(out bhErrorInfo);
systemErrorCode = bhErrorInfo.systemErrorCode;
systemErrorText = Marshal.PtrToStringAuto(bhErrorInfo.systemErrorText);
bhGiSErrorCode = new int[bhErrorInfo.countBHGiSError];
Marshal.Copy(bhErrorInfo.bhGiSErrorCode, bhGiSErrorCode, 0,
(int)bhErrorInfo.countBHGiSError);
BHImportFunctions.FreePtrMemory(bhErrorInfo.bhGiSErrorCode);
```

2.9.3 Функция GetBHGiSErrorStrings()

Заголовок объявления функции:

LPWSTR __declspec(dllimport) GetBHGiSErrorString(DWORD bhGiSErrorCode);

Входные параметры:

bhGiSErrorCode – код ошибки модуля

Возвращаемое значение:

Если передаваемый код ошибки соответствует существующему коду ошибки, возвращаемое значение – текстовое описание ошибки.
Если не соответствует, возвращаемое значение – NULL.

Пример вызова на языке C#:

```
var stringInfo = String.Empty;
var stringPtr = BHImportFunctions.GetBHGiSErrorString((uint)bhGiSErrorCode);
if (stringPtr != IntPtr.Zero)
    stringInfo = Marshal.PtrToStringAuto(stringPtr);
return stringInfo;
```

2.9.4 Класс BHGiSErrorClass

Переменные класса:

DWORD SystemErrorCode – системный код ошибки
LPWSTR SystemErrorText – описание системной ошибки
std::stack<DWORD> BHGiSErrorCode – коды ошибок BHGiSErrorCode

Открытые функции:

1) Конструктор

```
BHGiSErrorClass()
        :SystemErrorCode(0),
        SystemErrorText(nullptr)
{};
```

2) Деструктор

```
~BHGiSErrorClass()
{
        if(BHGiSErrorCode.size())
                BHGiSErrorCode.empty();
};
```

3) Установка ошибки GetLastError();

```
void SetSystemError(DWORD lastError);
```

4) Установка SEH ошибки

```
void SetSEHError(SE_Exception se_error);
```

5) Добавление информации об ошибке

```
void AddBHGiSErrorCode(DWORD BHGiSCode);
```

6) Возврат ошибки GetLastError();

```
DWORD GetSystemErrorCode();
```

7) Возврат текстового описания ошибки GetLastError();

```
LPWSTR GetSystemErrorText();
```

8) Возврат ошибки модуля

```
std::stack<DWORD> GetBHGiSErrorCode();
```

9) Возврат описания ошибки модуля

```
static LPWSTR GetGiSErrorInfo(DWORD errorCode)
```

Структура – информация об ошибке:

```
typedef struct BHErrorInfo
{
        DWORD systemErrorCode;
        LPWSTR systemErrorText;
        DWORD* bhGiSErrorCode;
        DWORD countBHGiSError;
public:
        BHErrorInfo()
                :systemErrorCode(0),
                systemErrorText(nullptr),
                bhGiSErrorCode(nullptr),
                countBHGiSError(0)
        {}
};
```

2.9.5 Класс ErrorRouting

Открытые функции:

1) Создание нового объекта BHGiSError
static void InitBHGiSError();
2) Уничтожение объекта BHGiSError
static void DeleteBHGiSError();
3) Добавление информации об ошибке
static void AddErrorInfo(DWORD BHGiScode);
4) Установка ошибки GetLastError();
static void SetSystemError(DWORD lastError);
5) Установка SEH ошибки
static void SetSEHError(SE_Exception se_error);
6) Возврат описания ошибки модуля
static LPWSTR GetGiSErrorInfo(DWORD errorCode);

3 Сообщения системному программисту

3.1 Коды ошибок

Коды ошибок, возвращаемые криптоплатформой «Litoria Crypto Platform», представлены в таблице 3.1.

Таблица 3.1. Коды ошибок

№	Имя ошибки	Номер ошибки	Текст ошибки
1.	INIT_CREATE_EDS_ERROR	1000	Ошибка инициализации данных для создания ЭП.
2.	INVALID_CERTIFICATE_BLOB	1001	Неверный блоб сертификата подписчика.
3.	NOT_FIND_CERT_IN_MY	1002	Сертификат подписчика отсутствует в хранилище "Личные".
4.	NOT_FIND_HASH_ALG	1003	Алгоритм хеширования не определен.
5.	SIGNING_CERTIFICATE_V2_ERROR	1004	Ошибка формирования атрибута SignerCertReferenceV2.
6.	SIGNING_CERTIFICATE_ERROR	1005	Ошибка формирования атрибута SignerCertReference.
7.	INIT_FUNC_PARAM_ERROR	1006	Ошибка инициализации параметров функции. Неверный параметр.
8.	NOT_FIND_CRYPT_ALG	1007	Подходящий криптографический алгоритм не определен.
9.	START_CACHING_CONTAINER_ERROR	1008	Ошибка в функции кэширования контейнера.
10.	END_CACHING_CONTAINER_ERROR	1009	Ошибка в функции отмены кэширования контейнера.
11.	CREATE_HASH_ERROR	2000	Ошибка вычисления хеш-значения.
12.	GET_APPROPRIATE_CP_ERROR	2001	Не удалось найти подходящий криптопровайдер.
13.	GET_HASH_ALG_ID_ERROR	2002	Не удалось получить подходящий алгоритм хеширования.
14.	ININ_HASH_FILE_ERROR	2003	Ошибка инициализации данных для хеширования файла.
15.	HASHING_FILE_ERROR	2004	Ошибка хеширования файла.
16.	GET_HASH_RESULT_ERROR	2005	Ошибка получения результата хеширования файла.
17.	HASH_ALG_ERROR	2006	Для инициализации механизма хеширования передан алгоритм, не являющийся алгоритмом хеширования.
18.	CREATE_EDS_ERROR	3000	Ошибка создания ЭП.
19.	SIGN_ALG_ERROR	3001	Для инициализации механизма подписи передан алгоритм, не являющийся алгоритмом подписи.
20.	GET_SIGN_PARAM_ERROR	3002	Ошибка определения параметров создания ЭП.
21.	JOIN_SIGN_HASH_ALG_ERROR	3003	Не удалось определить комбинированный алгоритм вычисления хеш-значения и подписи.
22.	FORMAT_EDS_UNKNOWN	3004	Формат создаваемой подписи задан некорректно.
23.	ADD_EDS_ERROR	4000	Ошибка добавления ЭП.

№	Имя ошибки	Номер ошибки	Текст ошибки
24.	GET_EXISTING_SIGNS_ERROR	4001	Не удалось получить значения подписей из сообщения.
25.	NULL_HMSG	4002	Неверный дескриптор сообщения.
26.	GET_NEW_SIGNER_INDEX_ERROR	4003	Ошибка получения индекса добавленной подписи.
27.	COORDINATE_EDS_ERROR	4004	Ошибка заверки подписи.
28.	INIT_ADD_EDS_BY_HASH_ERROR	4005	Не удалось сформировать данные для удаленной подписи хеш-значения.
29.	SEPARATE_DATA_SIGN_ERROR	4006	Разделение подписанного документа на составляющие не произведено.
30.	ADD_INTO_TO_SIGN_ERROR	4007	Не удалось обновить подпись необходимой дополнительной информацией.
31.	ADD_HASH_EDS_JOIN_ERROR	4008	Произошла ошибка в процессе объединения компонентов механизма добавления подписи через удаленную подпись хеш-значения.
32.	JOIN_SIGN_ERROR	4009	Ошибка объединения 2-х отделенных ЭП.
33.	SIGN_HASH_ERROR	4010	Ошибка подписи хеш-значения.
34.	FORMAT_CMS_ERROR	4011	Не удалось сформировать CMS.
35.	ADD_COORD_ARCHIVAL_SIGNATURE_ERROR	4012	Некорректно осуществлять заверку или добавление подписи, если сообщение содержит одну или несколько подписей с архивными штампами времени.
36.	UPGRADE_SIGNET_TO_ADVANCED_ERROR	5000	Ошибка усовершенствования ЭП.
37.	GET_SIGN_ONLY_ERROR	5001	Не удалось получить значение подписи.
38.	GET_SIGNER_CERT_ERROR	5002	Не удалось получить информацию о сертификате подписавшего лица.
39.	SEND_REQUEST_ERROR	5003	Не удалось получить информацию из сети.
40.	SIGNER_CERT_CHAIN_ERROR	5004	Цепочка сертификата подписавшего лица составлена некорректно.
41.	TSP_CERT_CHAIN_ERROR	5005	Цепочка сертификата TSA-оператора составлена некорректно.
42.	OCSP_CERT_CHAIN_ERROR	5006	Цепочка сертификата OCSP-оператора составлена некорректно.
43.	CREATE_REVOCATION_VALUES_ERROR	5007	Ошибка формирования атрибутов RevocationValues и CompleteRevocationRefs.
44.	CREATE_CERTIFICATE_VALUES_ERROR	5008	Ошибка формирования атрибутов CompleteCertificateRefs и CertificateValues.
45.	CHECK_PROOF_VALIDITY_ERROR	5009	Ошибка проверки используемых сертификатов на действительность относительно времени внешнего штампа.
46.	SIGNER_CERT_VERIFY_TIME_ERROR	5010	Один или несколько сертификатов цепочки сертификата подписавшего лица недействителен.
47.	TSP_CERT_VERIFY_TIME_ERROR	5011	Один или несколько сертификатов цепочки сертификата TSA-оператора недействителен.
48.	OCSP_CERT_VERIFY_TIME_ERROR	5012	Один или несколько сертификатов цепочки сертификата OCSP-оператора недействителен.

№	Имя ошибки	Номер ошибки	Текст ошибки
49.	ADD_UNSIGNED_ATTR_ERROR	5013	Ошибка добавления неподписанного атрибута в подпись.
50.	GET_RESULT_ERROR	5014	Ошибка получения результата операции.
51.	GET_CERT_REV_ERROR	5015	Ошибка получения информации об отзыве сертификата.
52.	GET_REV_REF_ERROR	5016	Ошибка формирования ссылки на значение отзыва.
53.	GET_CERT_REF_ERROR	5017	Ошибка формирования ссылки на значение сертификата.
54.	CREATE_XADES_MANIFEST_ERROR	5018	Ошибка создания манифеста XAdES.
55.	CREATE_XADES_RECEIPT_ERROR	5019	Ошибка создания квитанции XAdES.
56.	EXTRACT_RECEIPT_ERROR	5020	Ошибка извлечения квитанции.
57.	XADES_SIGN_ERROR	5021	Ошибка подписи XAdES.
58.	XADES_SIGNED_RECEIPT_ERROR	5022	Ошибка создания подписанной квитанции XAdES.
59.	ADD_RECEIPT_TO_SIGNATURE_ERROR	5023	Ошибка добавления подписанной квитанции в ЭП, на которую эта квитанция создавалась.
60.	VERIFY_XADES_ERROR	5024	Ошибка проверки подписи XAdES.
61.	VERIFY_MANIFEST_ERROR	5025	Ошибка проверки манифеста.
62.	GET_RECEIPT_ERROR	5026	Ошибка получения информации по квитанции.
63.	GET_SIGNATURE_CERT_ERROR	5027	Ошибка получения сертификата подписчика XAdES.
64.	ADD_SIGNATURE_NODE_TO_DOCUMENT_ERROR	5028	Ошибка добавления узла с готовой подписью в документ.
65.	CREATE_SOAP_ERROR_RESPONSE	5029	Не удалось создать SOAP конверт с технологическим сообщением об ошибке.
66.	ERROR_PARSE_XML_BUFFER_TO_DOM_OBJECT	5030	Ошибка перевода буфера xml данных в DOM объект.
67.	ERROR_SERIALIZE_SIGNATURE_TO_BUFFER	5031	Ошибка сериализации объекта подписи в буфер.
68.	ERROR_PARSE_RECEIPT_FROM_BUFFER	5032	Ошибка разбора блока данных квитанции из буфера.
69.	ERROR_PARSE_MANIFEST_FROM_BUFFER	5033	Ошибка разбора манифеста из буфера.
70.	ERROR_PARSE_QUALIFYING_PROPERIES_FROM_STREAM	5034	Ошибка разбора секции QualifyingProperties xml подписи из потока.
71.	ERROR_PARSE_SIGNATURE_FROM_STREAM	5035	Ошибка разбора подписи из потока.
72.	NO_SIGNER_CERTIFICATE	5036	В подписи отсутствует сертификат подписчика.
73.	TOO_MANY_CERTIFICATES_IN_SIGNATURE	5037	В подписи присутствует больше одного сертификата.
74.	ERROR_PARSING_INPUT_DOCUMENT	5038	Ошибка разбора входного документа.
75.	GET_IN_TIME_STAMP_ERROR	6000	Не удалось получить штамп времени на значение подписи или он некорректен.

№	Имя ошибки	Номер ошибки	Текст ошибки
76.	GET_OUT_TIME_STAMP_ERROR	6001	Не удалось получить штамп времени на доказательства подписи или он некорректен.
77.	TSP_RESPONSE_STATUS_ERROR	6002	Статус ответа службы штампов времени некорректен.
78.	CHECK_TSP_SIGNATURE_ERROR	6003	Полученный штамп времени содержит ошибки.
79.	TSP_CERT_ERROR	6004	Не удалось получить информацию о сертификате TSA-оператора.
80.	TSP_NEEDED_OID_ERROR	6005	Сертификат службы штампов времени не предназначен для установки отметки времени.
81.	VERIFY_TSP_SIGNATURE_ERROR	6006	Подпись штампа времени математически некорректна.
82.	GET_TIME_IN_TIME_STAMP_ERROR	6007	Ошибка получения времени создания внутреннего штампа времени.
83.	GET_TIME_OUT_TIME_STAMP_ERROR	6008	Ошибка получения времени создания внешнего штампа времени.
84.	UPGRADE_IN_TIME_STAMP_ERROR	6009	Не удалось собрать доказательства действительности внутреннего штампа времени.
85.	UNKNOWN_HASH_ALG_ERROR	6010	В запросе указан неизвестный системе алгоритм.
86.	NON_HASH_ALG_ERROR	6011	Указанный в запросе алгоритм не является алгоритмом хеширования.
87.	SIZE_HASH_VALUE_ERROR	6012	Указанный алгоритм поддерживается системой, однако длина хеш-значения не соответствует требуемой.
88.	CREATE_TSP_REQUEST_ERROR	6013	Ошибка создания TSP-запроса.
89.	NOT_SUPP_HASH_OID_ERROR	6014	Алгоритм хеширования, указанный в запросе, не поддерживается для обработки.
90.	GET_CRL_ERROR	7000	Необходимый список отзыва сертификатов не найден.
91.	GET_LOCAL_CRL_ERROR	7001	Ошибка получения списка отзыва из хранилища.
92.	DOWNLOAD_CRL_ERROR	7002	Ошибка получения основного списка отзыва из сети.
93.	CERT_CRL_LOCAL_ERROR	7003	Для проверки передан пустой сертификат.
94.	STORE_CA_ERROR	7004	Ошибка доступа к хранилищу промежуточных центров сертификации.
95.	STORE_ROOT_ERROR	7005	Ошибка доступа к хранилищу корневых центров сертификации.
96.	CERT_CRL_REVOKED	7006	Проверяемый сертификат отозван по списку отзыва.
97.	GET_CRL_URLS_ERROR	7007	Ошибка получения точек распространения списка отзыва.
98.	CRL_NOT_URLS	7008	Сертификат не содержит ни одной точки распространения основного списка отзыва.
99.	CRL_CDP_ERROR	7009	Не удалось определить формат точки распространения списка отзыва.

№	Имя ошибки	Номер ошибки	Текст ошибки
100.	CRL_TIME_VALIDITY_ERROR	7010	Полученный список отзыва недействительный по времени.
101.	ADD_CRL_TO_STORE_ERROR	7011	Не удалось установить список отзыва в хранилище сертификатов.
102.	CREATE_OCSP_REQUEST_ERROR	7012	Ошибка создания OCSP-запроса.
103.	GET_OCSP_URLS_ERROR	7013	Ошибка получения точек распространения OCSP.
104.	OCSP_NOT_URLS	7014	Сертификат не содержит ни одной точки распространения OCSP.
105.	OCSP_CDP_ERROR	7015	Не удалось определить формат точки распространения OCSP.
106.	GET_ISSUER_ERROR	7016	Поставщик сертификата не определен.
107.	GET_OCSP_RESPONSE_ERROR	7017	Не удалось получить OCSP-ответ.
108.	OCSP_CERT_ERROR	7018	Сертификат OCSP-оператора не определен.
109.	VERIFY_CERT_CHAIN_ERROR	7019	Цепочка проверяемого сертификата составлена некорректно.
110.	VERIFY_CERT_ISSUER_ERROR	7020	Издатель сертификата не определен, либо сертификат самоподписанный.
111.	VERIFY_CERT_BY_CRL_ERROR	7021	Не удалось проверить сертификат по списку отзыва.
112.	VERIFY_CERT_BY_OCSP_ERROR	7022	Не удалось проверить сертификат по OCSP.
113.	FORMATE_CRL_ERROR	7023	Не удалось сформировать список отзыва сертификатов.
114.	FILL_CRL_INFO_ERROR	7024	Ошибка формирования структуры информации списка отзыва сертификатов.
115.	SIGN_CRL_INFO_ERROR	7025	Ошибка подписи информации списка отзыва сертификатов.
116.	CERT_OCSP_REVOKED	7026	Проверяемый сертификат отозван по OCSP.
117.	CHECK_CRL_SIGN_ERROR	7028	Ошибка проверки целостности списка отзыва. Список отзыва поврежден или некорректен.
118.	GET_CRL_ISSUER_ERROR	7029	Издатель списка отзыва не определен.
119.	CRL_ISSUER_INFO_UNNKOWN	7030	В проверяемом списке отзыва сертификатов отсутствует информация об издателе.
120.	OCSP_RESPONSE_ERROR_STATUS	7031	Статус ответа некорректен.
121.	CHAIN_CERT_OCSP_OPERETOR_ERROR	7032	Ошибка построения цепочки сертификата OCSP-службы.
122.	CHECK_OCSP_RESP_SIGN_ERROR	7033	Ошибка проверки целостности OCSP-ответа.
123.	GET_MAIN_CRL_ERROR	7034	Основной список отзыва не определен.
124.	DOWNLOAD_DELTA_CRL_ERROR	7035	Ошибка получения разностного списка отзыва из сети.
125.	FIND_CRL_ERROR	7036	Ошибка поиска списков отзыва для проверяемого сертификата.

№	Имя ошибки	Номер ошибки	Текст ошибки
126.	GET_DELTA_CRL_ERROR	7037	Необходимый для проверки разностный список отзыва не определен.
127.	PARSE_OCSP_REQ_ERROR	7100	Ошибка разбора OCSP-запроса.
128.	VERIFY_OCSP_REQ_SIGN_ERROR	7101	Ошибка проверки целостности OCSP-запроса.
129.	INIT_CONVERT_ERROR	8000	Ошибка инициализации преобразования отделенной ЭП в присоединенную.
130.	CONVERT_ERROR	8001	Ошибка преобразования отделенной ЭП в присоединенную.
131.	WRITE_HEADER_ERROR	8002	Ошибка записи заголовка подписи в результирующий файл.
132.	WRITE_DATA_ERROR	8003	Ошибка записи данных в результирующий файл.
133.	WRITE_EDS_ERROR	8004	Ошибка записи данных подписи в результирующий файл.
134.	PROXY_NAME_PASS_ERROR	9000	Необходимые имя пользователя и пароль для прокси заданы не верно.
135.	SEND_HTTP_REQUEST_ERROR	9001	Ошибка сетевого обращения.
136.	VERIFY_TYPE_EDS_ERROR	10000	Ошибка определения типа подписи.
137.	INIT_VERIFY_EDS_ERROR	10001	Ошибка инициализации данных для проверки ЭП.
138.	VERIFY_EDS_VALUE_ERROR	10002	Ошибка проверки математической целостности подписи.
139.	GET_FILE_TYPE_ERROR	10003	Ошибка определения типа данных в файле.
140.	GET_MESSAGE_TYPE_ERROR	10004	Ошибка определения типа сообщения.
141.	VERIFY_SIGN_DATA_ERROR	10005	Ошибка проверки ЭП подписанного сообщения.
142.	GET_DATA_FROM_EDS_ERROR	10006	Невозможно получить исходные данные подписанного сообщения.
143.	SIGNED_IS_DETACHED	10007	Подпись отделенная.
144.	UNKNOWN_HASH_ALGS	10008	Ни один из алгоритмов хеширования, представленных в структуре подписанного файла, не поддерживается в системе.
145.	HASH_ALG_NOT_EXIST_SIGN_CONTENT	10009	Алгоритм хеширования не содержится в списке используемых алгоритмов подписанного сообщения.
146.	OPEN_KEY_FIND_ERROR	10010	Не определен сертификат открытого ключа.
147.	VERIFY_ADVANCED_DECODE_ATTRIB_ERROR	11000	Ошибка декодирования атрибутов усовершенствованной ЭП.
148.	VERIFY_ADVANCED_ENCODE_ATTRIB_ERROR	11001	Ошибка кодирования атрибутов усовершенствованной ЭП.
149.	VERIFY_ADVANCED_CERT_VALUES_ERROR	11002	Отсутствуют значения сертификатов в атрибуте CertificateValues.
150.	CREATE_STORE_ERROR	11003	Ошибка создания дополнительного хранилища сертификатов.

№	Имя ошибки	Номер ошибки	Текст ошибки
151.	VERIFY_ADVANCED_SIGNER_CERT_CHAIN_ERROR	11004	Ошибка проверки доказательств усовершенствованной ЭП для сертификата подписчика.
152.	SIGNER_CERT_CHAIN_TSP_TIME_ERROR	11005	Один или несколько сертификатов из цепочки сертификата подписчика являлись недействительными на момент получения внешнего штампа времени.
153.	INCLUDE_CERT_CHAIN_IN_ATTRIB_ERROR	11006	В атрибут подписи включены не все сертификаты из цепочки.
154.	CHECK_SIGNER_CERT_BY_CERT_REFS_V2_ERROR	11007	Целостность подписи нарушена. Возможно, произошла подмена сертификата подписчика.
155.	CHECK_CERT_REFS_ERROR	11008	Ссылки на значения сертификатов были собраны не для всех сертификатов из цепочки, либо они недействительны.
156.	FIND_REV_VALUE_ERROR	11009	Значения отзывов были собраны не для всех сертификатов из цепочки, либо они недействительны.
157.	VERIFY_ADVANCED_CERT_IS_REVOKED	11010	Проверяемый сертификат на момент создания усовершенствованной ЭП был отозван.
158.	VERIFY_ADVANCED_CREATE_CRL_ERROR	11011	Не удалось сформировать список отзыва.
159.	VERIFY_ADVANCED_TSP_CERT_CHAIN_ERROR	11012	Ошибка проверки доказательств усовершенствованной ЭП для сертификата TSA-оператора.
160.	TSP_CERT_CHAIN_NOW_TIME_ERROR	11013	Один или несколько сертификатов из цепочки сертификата TSA-оператора являются недействительными на текущий момент времени.
161.	VERIFY_ADVANCED_CHECK_REV_VALUES_ERROR	11014	Ошибка проверки корректности значений отзывов.
162.	GET_OCSP_TIME_THIS_UPDATE_ERROR	11015	Ошибка получения времени формирования OCSP-ответа.
163.	CHECK_OCSP_ERROR	11016	Ошибка проверки корректности OCSP-ответа.
164.	OCSP_CERT_NOCHECK_ERROR	11017	В сертификате OCSP отсутствует необходимое расширение szOID_PKIX_OCSP_NOCHECK.
165.	VERIFY_OCSP_SIGNATURE_ERROR	11018	Подпись OCSP-ответа математически некорректна.
166.	VERIFY_ADVANCED_CHECK_REV_REFS_ERROR	11019	Ссылки на значения отзывов были собраны не для всех значений отзывов, либо они недействительны.
167.	VERIFY_ADVANCED_OCSP_CERT_CHAIN_ERROR	11020	Ошибка проверки доказательств усовершенствованной ЭП для сертификата OCSP-оператора.
168.	OCSP_CERT_CHAIN_TSP_TIME_ERROR	11021	Один или несколько сертификатов из цепочки сертификата OCSP-оператора являлись недействительными на момент получения внешнего штампа времени.
169.	VERIFY_ADVANCED_IN_TSP_ERROR	11022	Ошибка проверки внутреннего штампа времени и соответствия хеш-значений.

№	Имя ошибки	Номер ошибки	Текст ошибки
170.	COMPARE_HASH_IN_TSP_ERROR	11023	Хеш-значение подписи не соответствует хеш-значению во внутреннем штампе времени.
171.	VERIFY_ADVANCED_OUT_TSP_ERROR	11024	Ошибка проверки внешнего штампа времени и соответствия хеш-значений.
172.	COMPARE_HASH_OUT_TSP_ERROR	11025	Хеш-значение доказательств подписи не соответствует хеш-значению во внешнем штампе времени.
173.	FIND_OCSP_CERT_ERROR	11026	Не удалось определить сертификат OCSP-оператора.
174.	SINHRON_OSCP_TSP_TIME_ERROR	11027	Время в OCSP-ответе опережает время во внешнем штампе. Сервера OCSP и TSP были не синхронизированы по времени.
175.	OCSP_CERT_KP_OCSP_SIGNING_ERROR	11028	Сертификат службы OCSP не предназначен для подписи OCSP-ответов.
176.	GET_SING_ATTR_ERROR	11029	Ошибка получения атрибута подписи.
177.	FIND_SIGN_ATTR_ERROR	11030	Атрибут с заданным типом отсутствует в подписи.
178.	ADD_SING_ATTR_ERROR	11031	Ошибка добавления атрибута в подпись.
179.	CERT_STATUS_UNKNOWN	11032	Статус проверяемого сертификата не определен.
180.	GET_SIGN_TSP_INFO_ERROR	11033	Ошибка получения информации о штампах времени внутри подписи.
181.	GET_CERT_PROOF_ERROR	11034	Ошибка получения информации об отзыве сертификата.
182.	GET_CRL_INFO_ERROR	11035	Ошибка получения информации о списке отзыва.
183.	FIND_REV_REF_ERROR	11036	Хеш-значение отзыва не найдено либо оно недействительно.
184.	INIT_ENCRYPT_ERROR	12000	Ошибка инициализации данных для шифрования сообщения.
185.	INVALID_ENCRYPT_CERTIFICATE_BLOB	12001	Неверный блоб сертификата получателя.
186.	INVALID_ENCRYPT_CERTIFICATE	12002	Сертификат получателя не определен.
187.	FIND_ENCRYPT_OID_ERROR	12003	Не найден подходящий алгоритм шифрования.
188.	FIND_SKEY_ERROR	12004	Не найден закрытый ключ, соответствующий сертификату открытого ключа.
189.	PIN_CODE_ERROR	12005	Набор ключей не существует. Возможно, вы ввели неверный пин код.
190.	INIT_DECRYPT_ERROR	12006	Ошибка инициализации данных для расшифровывания сообщения.
191.	DECRYPT_ERROR	12007	Ошибка расшифровывания сообщения.
192.	ENCRYPT_ERROR	12008	Ошибка шифрования сообщения.
193.	SECURE_DELETE_ERROR	12009	Ошибка гарантированного удаления данных.
194.	PROV_NAME_ERROR	13000	Криптопровайдер не указан.
195.	GET_PROV_TYPE_ERROR	13001	Не удалось определить тип криптопровайдера.
196.	CREATE_KEY_CONTAINER_ERROR	13002	Ошибка создания контейнера ключей.
197.	CREATE_KEY_PAIR_ERROR	13003	Ошибка генерации пары ключей.

№	Имя ошибки	Номер ошибки	Текст ошибки
198.	GET_PUBLIC_KEY_STRUCT_LEN_ERROR	13004	Ошибка получения длины структуры под экспортируемый открытый ключ.
199.	GET_PUBLIC_KEY_ERROR	13005	Ошибка экспортирования открытого ключа.
200.	ENCODE_SUBJECT_INFO_ERROR	13006	Ошибка кодирования информации о владельце сертификата.
201.	SIGN_ADN_ENCODE_CERT_REQ_ERROR	13007	Возникла ошибка в процессе подписания и кодирования запроса на сертификат.
202.	BASE64_ENCODE_ERROR	13008	Ошибка кодирования информации в base64.
203.	INVALID_USING_CERTIFICATE_BLOB	13009	Неверный блоб используемого сертификата.
204.	GET_HPROV_ERROR	13010	Не удалось получить контекст криптопровайдера.
205.	GET_PP_ENUMCONTAINERS_ERROR	13011	Ошибка при получении размера имени контейнера секретного ключа.
206.	GET_PROV_NAME_ERROR	13012	Ошибка при получении имени контейнера секретного ключа.
207.	GET_CONTAINER_NAME_ERROR	13013	Имя контейнера ключа не задано и не определено.
208.	SET_PRIVATE_KEY_FOR_CERT_ERROR	13014	Ошибка создания в сертификате ссылки на закрытый ключ.
209.	ADD_CERT_CONTEXT_TO_STORE_ERROR	13015	Не удалось установить сертификат в хранилище сертификатов.
210.	BASE64_DECODE_ERROR	13016	Ошибка декодирования информации из BASE64.
211.	CREATE_CERT_REQUEST_ERROR	13017	Ошибка создания запроса на сертификат.
212.	SET_PRIVATE_KEY_ERORR	13018	Ошибка установки сертификата и создания связки открытый-закрытый ключ.
213.	GET_CERT_FROM_STORE_ERROR	13019	Ошибка получения сертификата из хранилища.
214.	INIT_GET_CERT_LIST_FROM_STORE_ERROR	13020	Ошибка инициализации данных для получения списка сертификатов из заданного хранилища.
215.	GET_CERT_LIST_FROM_STORE_ERROR	13021	Ошибка получения списка сертификатов из заданного хранилища.
216.	GET_CERT_EXT_INFO_ERROR	13022	Ошибка получения расширенной информации о сертификате.
217.	GET_CERT_CHAIN_ERROR	13023	Не удалось получить цепочку сертификации для выбранного сертификата.
218.	CREATE_CERT_CHAIN_ERROR	13024	Цепочка сертификации составленная некорректно.
219.	INVALID_USING_CRL_BLOB	13025	Неверный блоб используемого списка отзыва сертификатов.
220.	ADD_CRL_CONTEXT_TO_STORE_ERROR	13026	Не удалось установить список отзыва сертификатов в хранилище.
221.	CREATE_SELF_SIGNED_CERT_ERROR	13027	Ошибка создания самоподписанного сертификата.
222.	FILL_SELF_CERT_INFO_ERROR	13028	Ошибка формирования структуры информации о самоподписанном сертификате.

№	Имя ошибки	Номер ошибки	Текст ошибки
223.	SIGN_SELF_CERT_ERROR	13029	Ошибка подписи информации о самоподписанном сертификате.
224.	CREATE_USER_CERT_ERROR	13030	Ошибка выпуска сертификата пользователя.
225.	DET_CP_KEY_CONTAINER_ERROR	13031	Ошибка определения информации для доступа к закрытому ключу.
226.	FILL_USER_CERT_INFO_ERROR	13032	Ошибка формирования структуры информации о пользовательском сертификате.
227.	SIGN_USER_CERT_ERROR	13033	Ошибка подписи информации о пользовательском сертификате.
228.	COMPARE_OPEN_KEYS_ERROR	13034	Соответствие открытого ключа устанавливаемого сертификата и открытого ключа в контейнере не установлено.
229.	CERT_REQUEST_DATA_ERROR	13035	Данные запроса на сертификат имеют некорректный формат.
230.	FILL_CERT_REQ_INFO_ERROR	13036	Ошибка формирования структуры информации о запросе на сертификат.
231.	GET_CERT_REQ_SIGN_VALUE_ERROR	13037	Не удалось рассчитать значение ЭП на данные запроса на сертификат.
232.	CERT_REQ_SIGN_VALUE_CORRETC_ERROR	13038	Значение подписи запроса на сертификат математически некорректно.
233.	GET_CERT_EXT_ERROR	13039	Не удалось получить информацию о расширениях сертификата.
234.	ADD_CERT_CONTEXT_TO_DEVICE_ERROR	13100	Не удалось установить сертификат на устройство.
235.	STRUCT_GET_ENCODE_ERROR	14000	Ошибка декодирования структуры данных.
236.	STRUCT_BER_DECODE_ERROR	14001	Ошибка кодирования информации в структуру данных.
237.	SET_OID_ERROR	14002	Ошибка декодирования объектного идентификатора.
238.	FILE_NAME_ERROR	14003	Имя файла не задано.
239.	CREATE_DVC_REQUEST_ERROR	15000	Ошибка создания DVCS-запроса.
240.	VERIFY_CERT_BLOB_ERROR	15001	Неверный блоб проверяемого сертификата.
241.	DVC_REQUEST_TYPE_ERROR	15002	Тип DVCS-запроса некорректен.
242.	CREATE_DVCS_SIGNATURE_ERROR	15003	Ошибка подписи данных протокола DVCS.
243.	VERIFY_DVC_SIGNATURE_ERROR	15004	Ошибка проверки подписи данных протокола DVCS.
244.	PARSE_DVC_ERROR	15005	Ошибка разбора DVC-квитанции.
245.	CREATE_DVC_RESPONSE_ERROR	15006	Ошибка создания DVC-квитанции.
246.	CHANGE_DVC_SERIAL_ERROR	15007	Ошибка замены значения серийного номера в DVC-квитанции.
247.	GET_CRYPT_ALG_FROM_DVCREQ_ERROR	15008	Ошибка получения криптографических алгоритмов из DVCS-запроса.

№	Имя ошибки	Номер ошибки	Текст ошибки
248.	PROLONGATION_DVC_ERROR	15009	Не удалось выполнить операцию пролонгации квитанции.
249.	GET_DVCS_TYPE_ERROR	15010	Ошибка определения типа DVCS-запроса.
250.	GET_DVC_REQ_INFO_ERROR	15011	Ошибка получения информации из структуры DVCS-запроса.
251.	GET_SUB_DVC_REQ_ERROR	15012	Ошибка получения "подзапроса" из структуры DVCS-запроса.
252.	DVC_REQ_INDEX_ERROR	15013	Индекс запрашиваемого компонента находится вне границ массива.
253.	GET_SIGNED_ALGS_ERROR	15014	Ошибка получения алгоритмов подписи из подписанного сообщения.
254.	MERGE_DVCS_RESP_ERROR	15015	Ошибка объединения нескольких DVCS-ответов в результирующий DVCS-ответ.
255.	INTEGRITY_DVCS_ERROR	15016	Ошибка проверки соответствия DVCS-ответа DVCS-запросу.
256.	NOT_INTEGRITY_DVCS_ERROR	15017	Ответ не соответствует запросу, произошла подмена ответа.
257.	DELETE_AUTH_INFO_ERROR	15018	Ошибка удаления аутентификационной информации из DVCS-запроса.
258.	CHECK_CORRESP_TYPE_RESP_ERROR	15019	Тип квитанции dvErrorNote не может быть сопоставлен с исходными данными.
259.	BH_CERT_TRUST_IS_NOT_TIME_VALID	16000	Сертификат или один из сертификатов в цепочке не действителен на текущий момент времени.
260.	BH_CERT_TRUST_IS_REVOKED	16001	Сертификат или один из сертификатов в цепочке отозван.
261.	BH_CERT_TRUST_IS_NOT_SIGNATURE_VALID	16002	Сертификат или один из сертификатов в цепочке искажен (содержит недействительную электронную подпись).
262.	BH_CERT_TRUST_IS_NOT_VALID_FOR_USAGE	16003	Сертификат или один из сертификатов в цепочке не предназначен для данного использования.
263.	BH_CERT_TRUST_IS_UNTRUSTED_ROOT	16004	Сертификат или один из сертификатов в цепочке издан недоверенным центром сертификации.
264.	BH_CERT_TRUST_IS_CYCLIC	16005	Один из сертификатов в цепочке издан центром, сертифицированным проверяемым сертификатом (циклическая цепочка).
265.	BH_CERT_TRUST_IS_PARTIAL_CHAIN	16006	Построение цепочки сертификации не завершено (неполная цепочка).
266.	BH_CERT_MEDI_CERT_CHECK_REVOKED_ERROR	16007	Ошибка проверки отзыва промежуточного сертификата цепочки.
267.	GET_CSP_NAMES_ERROR	17000	Ошибка получения имен криптопровайдеров в системе.
268.	GET_CSP_PARAMS_ERROR	17001	Ошибка получения параметров криптопровайдера.
269.	GET_CERT_FROM_CONTAINER_ERROR	17002	Ошибка получения сертификата ключа подписи из контейнера.

№	Имя ошибки	Номер ошибки	Текст ошибки
270.	DELETE_CONTAINER_ERROR	17003	Ошибка удаления контейнера ключей.
271.	ARCHIVAL_TIME_STAMPING_ERROR	18000	Ошибка простановки архивного штампа времени.
272.	VERIFY_SIGNATURE_ERROR	18001	Подпись недействительна.
273.	EDS_NOT_ADVANCED_ERROR	18002	Подпись не является усовершенствованной.
274.	FIND_LAST_TIME_STAMP_ERROR	18003	Заверяющий подпись штамп времени неопределен.
275.	UPGRADE_LAST_TIME_STAMP_ERROR	18004	Не удалось собрать доказательства действительности заверяющего штампа времени.
276.	REPLACE_LAST_TIME_STAMP_ERROR	18005	Ошибка замены заверяющего штампа в подписи.
277.	CALC_HASH_FOR_ARCHIVAL_TIME_STAMP_ERROR	18006	Не удалось рассчитать хеш-значение для формирования запроса в TSP-службу.
278.	CREATE_ARCHIVE_TIME_ADD_TO_SIGNATURE	18007	Не удалось сформировать архивный штамп времени и добавить его в обрабатываемую подпись.
279.	EDS_ALREADY_ADVANCED_ERROR	18008	Подпись уже является усовершенствованной.
280.	ATS_VERIFY_SEPARATE_ERROR	18009	Не удалось проверить исходную подпись, либо разделить на подпись и данные.
281.	VERIFY_ARCHIVAL_TIME_STAMPS_ERROR	19000	Ошибка проверка корректности архивных штампов времени.
282.	GET_DATA_FROM_SIGN_FILE_ERROR	19001	Не удалось получить данные из файла-подписи.
283.	VERIFY_ADVANCED_OUT_TSP_CERT_CHAIN_ERROR	19002	Ошибка проверки наличия и корректности доказательств действительности цепочки сертификата TSA-оператора внешнего штампа времени на момент формирования первого архивного штампа.
284.	CALC_HASH_FOR_VERIFY_ARCHIVAL_TS_ERROR	19003	Не удалось рассчитать хеш-значение для проверки корректности архивного штампа.
285.	COMPARE_HASH_ARCHIVAL_TIME_SATMP_ERROR	19004	Хеш-значение в архивном штампе времени не соответствует вновь вычисленному.
286.	VERIFY_ADVANCED_ARCHIVAL_TSP_CERT_CHAIN_ERROR	19005	Ошибка проверки наличия и корректности доказательств действительности цепочки сертификата TSA-оператора архивного штампа времени на момент формирования последующего архивного штампа.
287.	LAST_ARCHIVAL_TSP_CERT_CHAIN_ERROR	19006	Цепочка сертификата TSA-оператора последнего архивного штампа времени составлена некорректно.
288.	CHECK_LAST_ARCHIVAL_TSP_SIGNATURE_ERROR	19007	Ошибка проверка корректности последнего архивного штампа.
289.	GET_CERT_REQ_EXT_ERROR	20000	Не удалось получить расширения сертификата из запроса на сертификат.

№	Имя ошибки	Номер ошибки	Текст ошибки
290.	INCORRECT_EKU_EXT_ERROR	20001	Расширение EnhancedKeyUsage имеет некорректный формат.
291.	INCORRECT_KU_EXT_ERROR	20002	Расширение KeyUsage имеет некорректный формат.
292.	FORMATE_CERT_EXTENSIONS_ERROR	20003	Произошла ошибка в процессе формирования asn.1 структур расширений сертификата.
293.	INCORRECT_EXTENSION_PARAM_ERROR	20004	Параметры расширения заданы неверно.
294.	FORMATE_EKU_EXT_ERROR	20005	Ошибка формирования расширения EnhancedKeyUsage.
295.	DECODE_ENHANCE_ERROR	20006	Ошибка кодирования одной из целей использования сертификата.
296.	FORMATE_KU_EXT_ERROR	20007	Ошибка формирования расширения KeyUsage.
297.	FORMATE_AKI_EXT_ERROR	20008	Ошибка формирования расширения AuthorityKeyIdentifier.
298.	FORMATE_SKI_EXT_ERROR	20009	Ошибка формирования расширения SubjectKeyIdentifier.
299.	FORMATE_BAS_CONSTR_EXT_ERROR	20010	Ошибка формирования расширения BasicConstraints.
300.	FORMATE_CRL_REV_RES_EXT_ERROR	20011	Ошибка формирования расширения CRLRevokeReason.
301.	FORMATE_CRL_NUMBER_EXT_ERROR	20012	Ошибка формирования расширения CRLNumber.
302.	FORMATE_CRL_DISTR_POINTS_ERROR	20013	Ошибка формирования расширения CRLDistributionPoints.
303.	INCORRECT_CRL_DP_EXT_ERROR	20014	Расширение CRLDistributionPoints имеет некорректный формат.
304.	FORMATE_AUTH_ACCESS_EXT_ERROR	20015	Ошибка формирования расширения AuthorityInfoAccessSyntax.
305.	INCORRECT_AUTH_INFO_ACCESS_EXT_ERROR	20016	Расширение AuthorityInfoAccessSyntax имеет некорректный формат.
306.	FORMATE_CERT_POLICIES_EXT_ERROR	20017	Ошибка формирования расширения CertificatePolicies.
307.	INCORRECT_CERT_POLICIES_EXT_ERROR	20018	Расширение CertificatePolicies имеет некорректный формат.
308.	FORMATE_SUB_SIGN_TOOL_EXT_ERROR	20019	Ошибка формирования расширения SubjectSignTool.
309.	INCORRECT_SUB_SIGN_TOOL_EXT_ERROR	20020	Расширение SubjectSignTool имеет некорректный формат.
310.	FORMATE_ISSUER_SIGN_TOOL_EXT_ERROR	20021	Ошибка формирования расширения IssuerSignTool.
311.	INCORRECT_ISSUER_SIGN_TOOL_EXT_ERROR	20022	Расширение IssuerSignTool имеет некорректный формат.
312.	FORMATE_SUB_ALT_EXT_ERROR	20023	Ошибка формирования расширения SubjectAltName.
313.	INCORRECT_SUB_ALT_NAME_EXT_ERROR	20024	Расширение SubjectAltName имеет некорректный формат.
314.	CREATE_ATTR_CERT_ERROR	21000	Ошибка выпуска атрибутного сертификата.

№	Имя ошибки	Номер ошибки	Текст ошибки
315.	FORMATE_ATTRCERT_ATTRS_ERROR	21001	Произошла ошибка в процессе формирования asn.1 структур привилегий атрибутного сертификата.
316.	FORMATE_ROLE_ATTR_ERROR	21002	Ошибка формирования атрибута RoleSyntax.
317.	CREATE_CERT_REVOKE_REQ_ERROR	22000	Ошибка создания запроса на отзыв сертификата.
318.	PARSE_CERT_REVOKE_REQ_ERROR	22001	Ошибка разбора запроса на отзыв сертификата.
319.	CREATE_CERT_REVOKE_RESP_ERROR	22002	Ошибка создания ответа на запрос на отзыв сертификата.
320.	BH_TOKEN_NOT_FOUND	70000	Устройство не найдено. ???
321.	BH_INITIALIZE_ERROR	70001	Ошибка инициализации.
322.	BH_GET_SLOT_LIST_ERROR	70002	Ошибка получения списка слотов.
323.	BH_OPEN_SESSION_ERROR	70003	Ошибка открытия сессии.
324.	BH_LOGIN_ERROR	70004	Ошибка авторизации.
325.	BH_LOGOUT_ERROR	70005	Ошибка вылогирования.
326.	BH_DIGIST_INIT_ERROR	70006	Ошибка инициализации хеширования.
327.	BH_DIGIST_ERROR	70007	Ошибка хеширования.
328.	BH_DIGIST_UPDATE_ERROR	70008	Ошибка хеширования; поток, обновление.
329.	BH_DIGIST_FINAL_ERROR	70009	Ошибка при получении значения хеш-значения.
330.	BH_SIGN_INIT_ERROR	70010	Ошибка инициализации создания подписи.
331.	BH_SIGN_ERROR	70011	Ошибка в процессе создания подписи.
332.	BH_VERIFY_INIT_ERROR	70012	Ошибка инициализации проверки подписи.
333.	BH_VERIFY_ERROR	70013	Ошибка в процессе проверки подписи.
334.	BH_FIND_OBJECTS_INIT_ERROR	70014	Ошибка инициализации поиска объектов.
335.	BH_FIND_OBJECTS_ERROR	70015	Ошибка поиска объектов.
336.	BH_FIND_OBJECTS_FINAL_ERROR	70016	Ошибка при получении результата поиска объектов.
337.	BH_GET_OBJECT_ATTRIBUTE_VALUE_ERROR	70017	Ошибка при получении значения атрибута объекта на устройстве.
338.	BH_SET_OBJECT_ATTRIBUTE_VALUE_ERROR	70018	Ошибка при установке значения атрибута объекта на устройстве.
339.	BH_PKCS7_SIGN_ERROR	70019	Ошибка создания CMS сообщения.
340.	BH_PKCS7_VERIFY_ERROR	70020	Ошибка проверки CMS сообщения.
341.	BH_RANDOM_GENERATION_ERROR	70021	Ошибка генерации случайного числа аппаратными средствами устройства.
342.	BH_GENERATE_KEY_PAIR_ERROR	70022	Ошибка при генерации ключевой пары на устройстве.
343.	BH_GENERATE_SESSION_KEY_ERROR	70023	Ошибка при генерации сессионного ключа на устройстве.
344.	BH_DERIVE_KEY_ERROR	70024	Ошибка создания ключа согласования на токене.
345.	BH_WRAP_KEY_ERROR	70025	Ошибка экспорта сессионного ключа на ключе согласовании.

№	Имя ошибки	Номер ошибки	Текст ошибки
346.	BH_UNWRAP_KEY_ERROR	70026	Ошибка импорта сессионного ключа на ключе согласовании.
347.	BH_ENCRYPT_INIT_ERROR	70027	Ошибка инициализации зашифровывания данных.
348.	BH_ENCRYPT_ERROR	70028	Ошибка потокового зашифровывания.
349.	BH_ENCRYPT_FINAL_ERROR	70029	Ошибка завершения операции зашифровывания данных.
350.	BH_DECRYPT_INIT_ERROR	70030	Ошибка инициализации расшифровывания данных.
351.	BH_DECRYPT_ERROR	70031	Ошибка потокового расшифровывания.
352.	BH_DECRYPT_FINAL_ERROR	70032	Ошибка завершения операции расшифровывания.
353.	BH_CREATE_OBJECT_ERROR	70033	Ошибка при создании объекта на устройстве.
354.	BH_DESTROY_OBJECT_ERRO R	70034	Ошибка удаления объекта с устройства.
355.	BH_CREATE_CSR_ERROR	70035	Ошибка создания запроса на сертификат.
356.	BH_GET_MECHANISM_LIST_ ERROR	70036	Ошибка получения списка механизмов.
357.	BH_GET_MECHANISM_INFO_ ERROR	70037	Ошибка получения информации о механизме.
358.	COMPARE_DOCUMENT_HAS H_ERROR	70038	Ошибка сравнения хеш-значений документов.
359.	VERIFY_EDS_SIGNATURE_E RROR	70039	Ошибка при проверке сигнатуры подписи.
360.	BH_DLL_ALREADY_ITITIALI ZED	70040	Модуль pkcs11 уже проинициализирован.
361.	BH_DECODE_STRUCT_ERRO R	70041	Ошибка декодирования в структуру данных.
362.	BH_ENCODE_STRUCT_ERRO R	70042	Ошибка при кодировании структуры данных.
363.	BH_GET_STRING_ERROR	70043	Ошибка при получении строчки.
364.	BH_GET_OID_ERROR	70044	Ошибка при получении ОИД-а.
365.	BH_NO_TOKEN_KEYS_ERRO R	70045	Отсутствуют ключи на устройстве.
366.	BH_UNKNOW_STORE_ERRO R	70046	Неизвестное имя хранилища.
367.	BH_PARAM_ERROR	70047	Функция вызвана с некорректными параметрами.
368.	BH_CANT_FIND_PRIVATE_K EY_ERROR	70048	На устройстве отсутствует необходимый закрытый ключ.
369.	BH_CANT_FIND_PUBLIC_KE Y_ERROR	70049	На устройстве отсутствует необходимый открытый ключ.
370.	BH_GET_SLOT_INFO_ERROR	70050	Ошибка получения информации о слоте.
371.	BH_GET_TOKEN_INFO_ERRO R	70051	Ошибка при получении информации об устройстве.
372.	BH_CANT_LOAD_LIBRARY_ ERROR	70052	Невозможно загрузить библиотеку pkcs11.

№	Имя ошибки	Номер ошибки	Текст ошибки
373.	BH_NO_PIN_CODE_PROVIDED	70053	Функции не передан пин код к устройству.
374.	BH_CKR_HOST_MEMORY	71001	Недостаточно памяти для выполнения функции.
375.	BH_CKR_SLOT_ID_INVALID	71002	Неправильный идентификатор слота.
376.	BH_CKR_GENERAL_ERROR	71003	Критическая ошибка, связанная с аппаратным обеспечением.
377.	BH_CKR_FUNCTION_FAILED	71004	При выполнении функции возник сбой.
378.	BH_CKR_ARGUMENTS_BAD	71005	Недопустимый аргумент.
379.	BH_CKR_ATTRIBUTE_READ_ONLY	71006	Предпринята попытка присвоения значения атрибуту, который нельзя изменять.
380.	BH_CKR_CANT_LOCK	71007	Приложение не поддерживает защиту потоков от одновременного использования различными функциями.
381.	BH_CKR_ATTRIBUTE_SENSITIVE	71008	Запрашиваемый атрибут недоступен для чтения.
382.	BH_CKR_ATTRIBUTE_TYPE_INVALID	71009	Некорректный тип атрибута.
383.	BH_CKR_ATTRIBUTE_VALUE_INVALID	71010	Атрибут нулевой длины.
384.	BH_CKR_DEVICE_ERROR	71011	Ошибка при обращении к устройству или слоту.
385.	BH_CKR_DEVICE_MEMORY	71012	Для выполнения функции недостаточно памяти в устройстве.
386.	BH_CKR_DEVICE_REMOVED	71013	При выполнении функции устройство было отключено.
387.	BH_CKR_ENCRYPTED_DATA_LEN_RANGE	71014	Выбранный механизм не поддерживает расшифровывание данных выбранной длины.
388.	BH_CKR_FUNCTION_NOT_SUPPORTED	71015	Выбранная функция не поддерживается модулем сопряжения или не найден дополнительный подключаемый модуль.
389.	BH_CKR_KEY_HANDLE_INVALID	71016	Функции передан некорректный дескриптор ключа.
390.	BH_CKR_KEY_SIZE_RANGE	71017	Недопустимый размер ключа.
391.	BH_CKR_KEY_TYPE_INCONSISTENT	71018	Данный тип ключа не может использоваться с заданным механизмом.
392.	BH_CKR_KEY_FUNCTION_NOT_PERMITTED	71019	Ключ, используемый для выполнения криптографических операций, имеет атрибуты, не позволяющие использовать его для данных операций.
393.	BH_CKR_KEY_NOT_WRAPPABLE	71020	Ключ не может быть экспортирован.
394.	BH_CKR_MECHANISM_INVALID	71021	При выполнении криптографической функции был указан неправильный механизм.
395.	BH_CKR_MECHANISM_PARAM_INVALID	71022	При выполнении криптографической функции были заданы некорректные параметры механизма.
396.	BH_CKR_OBJECT_HANDLE_INVALID	71023	Функции передан некорректный дескриптор объекта.

№	Имя ошибки	Номер ошибки	Текст ошибки
397.	BH_CKR_OPERATION_ACTIVE	71024	Одна или несколько выполняющихся операций препятствуют выполнению новой операции.
398.	BH_CKR_OPERATION_NOT_INITIALIZED	71025	Выполнение операции без предварительного указания параметров невозможно.
399.	BH_CKR_PIN_INCORRECT	71026	Функции передан неверный пин код.
400.	BH_CKR_PIN_INVALID	71027	Значение пин кода содержит недопустимые символы.
401.	BH_CKR_PIN_LEN_RANGE	71028	Недопустимая длина пин кода.
402.	BH_CKR_PIN_LOCKED	71029	Пин код заблокирован.
403.	BH_CKR_SESSION_CLOSED	71030	При выполнении функции сеанс был закрыт.
404.	BH_CKR_SESSION_COUNT	71031	Достигнуто предельное количество открытых сеансов для данного устройства.
405.	BH_CKR_SESSION_HANDLE_INVALID	71032	Функции передан некорректный дескриптор сеанса.
406.	BH_CKR_SESSION_PARALLEL_NOT_SUPPORTED	71033	Невозможно открыть параллельный сеанс.
407.	BH_CKR_SESSION_EXISTS	71034	Открыт сеанс работы с тем же устройством. Поэтому устройство не может быть инициализировано с помощью функции C_InitToken.
408.	BH_CKR_SESSION_READ_ONLY_EXISTS	71035	Открыт сеанс только для чтения. Смена режима невозможна.
409.	BH_CKR_SESSION_READ_WRITE_SO_EXISTS	71036	Открыт сеанс чтения/записи. Открыть сеанс только для чтения невозможно.
410.	BH_CKR_SIGNATURE_INVALID	71037	Неправильное значение ЭП или имитовставки.
411.	BH_CKR_SIGNATURE_LEN_RANGE	71038	Значение ЭП неверно по крайней мере по длине.
412.	BH_CKR_TEMPLATE_INCOMPLETE	71039	Для создания объекта недостаточно атрибутов.
413.	BH_CKR_TEMPLATE_INCONSISTENT	71040	Заданные в шаблоне значения атрибутов противоречат друг другу.
414.	BH_CKR_TOKEN_NOT_PRESENT	71041	В момент выполнения функции устройство было отключено.
415.	BH_CKR_TOKEN_NOT_RECOGNIZED	71042	Невозможно определить тип устройства подключенного к слоту.
416.	BH_CKR_TOKEN_WRITE_PROTECTED	71043	Устройство недоступно для записи.
417.	BH_CKR_USER_ALREADY_LOGGED_IN	71044	Устройство уже работает в режиме, включение которого требует ввода данного пин кода.
418.	BH_CKR_USER_NOT_LOGGED_IN	71045	Неверный режим работы устройства.
419.	BH_CKR_USER_PIN_NOT_INITIALIZED	71046	Начальное значение пин кода не установлено.
420.	BH_CKR_USER_TYPE_INVALID	71047	Задан некорректный режим работы с модулем сопряжения.

№	Имя ошибки	Номер ошибки	Текст ошибки
421.	BH_CKR_USER_ANOTHER_ALREADY_LOGGED_IN	71048	Переключение из режима администратора в режим пользователя или из режима пользователя в режим администратора невозможно.
422.	BH_CKR_USER_TOO_MANY_TYPES	71049	Включение одновременно режимов пользователя и администратора невозможно.
423.	BH_CKR_BUFFER_TOO_SMALL	71050	Размер заданного буфера является недостаточным для сохранения результатов функции.
424.	BH_CKR_INFORMATION_SENSITIVE	71051	Запрашиваемый объект недоступен для чтения.
425.	BH_CKR_CRYPTOKI_NOT_INITIALIZED	71052	Выполнение функции без инициализации модуля сопряжения невозможно.
426.	BH_CKR_CRYPTOKI_ALREADY_INITIALIZED	71053	Повторная попытка инициализировать модуль сопряжения без предварительного выполнения функции C_Finalize.
427.	BH_CKR_FUNCTION_REJECTED	71054	Запрос ЭП отклонён пользователем.
428.	BH_SOAP_INFO_ERROR	71055	Ошибка в заполнении структуры soap параметров
429.	BH_SOAP_CREATE_ERROR	71056	Ошибка при создании soap конверта

Список сокращений

AES – Advanced Encryption Standard
CCPD – Certification of Claim of Possession of Data
CDP – CRL Distribution Points
CMS – Cryptographic Message Syntax
CPD – Certification of Possession of Data
CRL – Certificate Revocation List
CSP – Cryptographic Service Provider
DVC – Data Validation and Certification
DVCS – Data Validation and Certification Server
EDS – Electronic Digital Signature
HDD – Hard Disk Drive
LSB – Linux Standard Base
MS – MicroSoft
OCSP – Online Certificate Status Protocol
OID – Object Identifier
RAM – Random Access Memory
RFC – Request For Comments
RSA – аббревиатура криптографического алгоритма от фамилий Rivest, Shamir и Adleman
SVGA – Super Video Graphics Array
TSA – Time Stamping Authority
TSP – Time-Stamp Protocol
USB – Universal Serial Bus
VKPC – Validation of Public Key Certificates
VSD – Validation of digitally Signed Document
ГОСТ – ГОсударственный СТандарт
ДТС – Доверенная Третья Сторона
ИОК – Инфраструктура Открытых Ключей
ОС – Операционная Система
РФ – Российская Федерация
УЭП – Усовершенствованная ЭП
ЭП – Электронная Подпись