Криптографический интерфейс ViPNet CSP 4.2

Руководство разработчика

infotecs

ФРКЕ.00106-04 33 01

ОАО «ИнфоТеКС»

127287, г. Москва, Старый Петровско-Разумовский проезд, дом 1/23, строение 1

Тел: (495) 737-61-96 (горячая линия), 737-61-92, факс 737-72-78

Сайт компании «ИнфоТеКС»: http://www.infotecs.ru

Электронный адрес службы поддержки: hotline@infotecs.ru

Содержание

Введение

О документе

Документ содержит описание структур, констант и функций криптопровайдера ViPNet CSP, входящего в программные продукты ViPNet CSP и ViPNet CSP Linux. Криптопровайдер предназначен для встраивания криптографических функций в сторонние программные интерфейсы.

Для кого предназначен документ

Документ предназначен для разработчиков программного обеспечения в области криптографического преобразования данных.

Соглашения документа

Ниже перечислены соглашения, принятые в этом документе для выделения информации.

Таблица 1. Обозначения, используемые в примечаниях

Обозначение	Описание
	**Внимание!** Указывает на обязательное для исполнения или следования действие или информацию.
	**Примечание.** Указывает на необязательное, но желательное для исполнения или следования действие или информацию.
	**Совет.** Содержит дополнительную информацию общего характера.

Таблица 2. Обозначения, используемые для выделения информации в тексте

Обозначение	Описание
Название	Название элемента интерфейса. Например, заголовок окна, название поля, кнопки или клавиши.
Клавиша+Клавиша	Сочетание клавиш. Чтобы использовать сочетание клавиш, следует нажать первую клавишу и, не отпуская ее, нажать вторую клавишу.
Меню > Подменю > Команда	Иерархическая последовательность элементов. Например, пункты меню или разделы на панели навигации.
`Код`	Имя файла, путь, фрагмент текстового файла (кода) или команда, выполняемая из командной строки.

Об интерфейсе ViPNet CSP

Интерфейс ViPNet CSP — один из криптографических интерфейсов, реализованных в продуктах ViPNet CSP и ViPNet CSP Linux.

В операционной системе Windows интерфейс ViPNet CSP обеспечивает вызов криптографических функций из различных приложений ViPNet, Microsoft и другого ПО, использующего интерфейс CryptoAPI 2.0.

В операционной системе Linux интерфейс ViPNet CSP обеспечивает вызов криптографических функций из приложений, использующих интерфейс ViPNet CryptoAPI for Linux (см. документ «ViPNet CryptoAPI for Linux. Руководство разработчика»).

Назначение, область применения

Криптографический интерфейс ViPNet CSP (криптопровайдер ViPNet CSP) создан в соответствии с криптографическим интерфейсом фирмы Microsoft — Cryptographic Service Provider (CSP).

Криптографический интерфейс ViPNet CSP обеспечивает следующие функции:

- Создание ключей электронной подписи по алгоритмам ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012.
- Вычисление и проверку электронной подписи по алгоритмам ГОСТ Р 34.10-2001 и ГОСТ Р 34.10-2012.
- Хэширование данных в соответствии с алгоритмами ГОСТ Р 34.11-94 и ГОСТ Р 34.11-2012.
- Шифрование и имитозащиту данных в соответствии с алгоритмом ГОСТ 28147-89 в режимах гаммирования (OFB), гаммирования с обратной связью (CFB) и простой замены с зацеплением (CBC).
- Поддержку различных устройств хранения ключей (токенов, смарт-карт и других).
- Хранение сертификатов открытых ключей непосредственно в контейнере ключей.

Криптопровайдер ViPNet CSP поддерживает параметры и идентификацию алгоритмов и параметров в соответствии с RFC4357. В ОС Windows это обеспечивает совместимую работу с криптопровайдерами других производителей («КриптоПро», «СигналКом»).

В состав криптопровайдера ViPNet CSP входят дополнительные модули, обеспечивающие при работе под управлением ОС Windows вызов криптографических функций провайдера через интерфейс Microsoft CryptoAPI 2.0. Это позволяет обеспечить вызов криптографических функций из внешних приложений, например из стандартных приложений Microsoft.

Системные требования

Для успешной работы ViPNet CSP компьютер должен удовлетворять следующим требованиям:

- Объем оперативной памяти — не менее 512 Мбайт.
- Свободное место на жестком диске — 100 Мбайт.
- Для ОС Windows:
  - Процессор — Intel Core 2 Duo или другой схожий по производительности x86-совместимый процессор с количеством ядер 2 и более.
  - Операционная система:
    - Windows 7 — 32/64-разрядная, сборка 6.1.7601;
    - Windows Server 2008 R2 — 64-разрядная, сборка 6.1.7601;
    - Windows 8 — 32/64-разрядная, сборка 6.2.9200;
    - Windows Server 2012 — 64-разрядная, сборка 6.2.9200;
    - Windows 8.1 — 32/64-разрядная, сборка 6.3.9600;
    - Windows Server 2012 R2 — 64-разрядная, сборка 6.3.9600;
    - Windows 10 — 32/64-разрядная следующих версий и сборок: версия 1507 (сборка 10240), версия 1511 (сборка 10586), версия 1607 (сборка 14393), версия 1703 (сборка 15063), версия 1709 (сборка 16299), версия 1803 (сборка 17134).

    Для каждой из указанных сборок должны быть установлены последние пакеты обновлений. Работа ViPNet CSP на компьютерах, работающих под управлением операционных систем других сборок, не гарантируется.

    

    Примечание. В ОС Windows 10 поддерживаются все заявленные криптографические операции, кроме организации защищенных подключений по протоколу TLS в веб-браузере Microsoft Edge.

  - Internet Explorer — версия 8.0 или выше.
  - При использовании программ Microsoft Office — версия 2007, 2010, 2013.
- Для ОС Linux:
  - Поддерживаются следующие дистрибутивы (32- и 64-разрядные):
    - Гослинукс 6, 6.4;
    - Astra Linux 1.4, 1,5;
    - CentOS 4.7, 5.2, 6.0;
    - Debian 4.0r0, 4.0r9, 5.0.0, 5.0.8, 6.0.0, 6.0.3, 7.0.0;
    - Fedora 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18;

- Mandriva 2006.0, 2007.0, 2007.1, 2008.0, 2008.1, 2009.0, 2009.1, 2010.0, 2010.1, 2010.2, 2011, Corporate Server 4;
- openSUSE 10.2, 10.3, 11.0, 11.1, 11.2, 11.3, 11.4, 12.1, 12.2, 12.3;
- Oracle Linux 5, 5.3, 6;
- RHEL 4, 4.8, 5, 5.7, 6, 6.2;
- SLES 9 SP4, 10, 10 SP1, 10 SP2, 10 SP3, 10 SP4, 11, 11 SP1, 11 SP2;
- Ubuntu 7.04, 7.10, 8.04, 8.10, 9.04, 9.10, 10.04, 10.10, 11.04, 11.10, 12.04, 12.10, 13.04, 13.10, 14.04, 14.10.

ViPNet CSP поддерживает работу с несколькими типами электронных ключей. Подробную информацию о поддерживаемых электронных ключах см. в документах «ViPNet CSP. Руководство пользователя», «ViPNet CSP for Linux. Руководство пользователя».

Обратная связь

Дополнительная информация

Сведения о продуктах и решениях ViPNet, распространенные вопросы и другая полезная информация собраны на сайте ОАО «ИнфоТеКС»:

- Веб-портал документации ViPNet http://docs.infotecs.ru.
- Описание продуктов ViPNet http://www.infotecs.ru/products/line/.
- Информация о решениях ViPNet http://www.infotecs.ru/solutions/.
- Сборник часто задаваемых вопросов (FAQ) http://www.infotecs.ru/support/faq/.
- Форум пользователей продуктов ViPNet http://www.infotecs.ru/forum.
- Законодательная база в сфере защиты информации http://www.infotecs.ru/laws/.

Контактная информация

С вопросами по использованию продуктов ViPNet, пожеланиями или предложениями свяжитесь со специалистами ОАО «ИнфоТеКС». Для решения возникающих проблем обратитесь в службу технической поддержки.

- Техническая поддержка для пользователей продуктов ViPNet: hotline@infotecs.ru.
- Форма запроса в службу технической поддержки http://www.infotecs.ru/support/request/.
- Регистрация продуктов и консультации по телефону для клиентов, имеющих расширенный уровень технического сопровождения:

  8 (495) 737-6192,

  8 (800) 250-0260 — бесплатный звонок из любого региона России (кроме Москвы).

Распространение информации об уязвимостях продуктов ОАО «ИнфоТеКС» регулируется политикой ответственного разглашения http://infotecs.ru/products/disclosure.php. Если вы обнаружили уязвимости в продуктах компании, сообщите о них по адресу security-notifications@infotecs.ru.

1

Структуры и константы

Константы, символьные идентификаторы алгоритмов

Приведенные в таблице идентификаторы алгоритмов и параметров соответствуют RFC4357.

Таблица 3. Идентификаторы алгоритмов и параметров

Константа	Значение	Описание
VPN_DEF_PROV	Infotecs Cryptographic Service Provider	Имя провайдера ИнфоТеКС
VPN_DEF_PROV_2012_512	Infotecs GOST 2012/512 Cryptographic Service Provider	Имя провайдера ИнфоТеКС на основе алгоритма ГОСТ Р 34.10-2012 с длиной ключа 256 бит
VPN_DEF_PROV_2012_1024	Infotecs GOST 2012/1024 Cryptographic Service Provider	Имя провайдера ИнфоТеКС на основе алгоритма ГОСТ Р 34.10-2012 с длиной ключа 512 бит
VPN_PROV_TYPE	2	Тип провайдера на основе алгоритма ГОСТ Р 34.10-2001
VPN_PROV_TYPE_2012_512	77	Тип провайдера на основе алгоритма ГОСТ Р 34.10-2012 с длиной ключа 256 бит
VPN_PROV_TYPE_2012_1024	78	Тип провайдера на основе алгоритма ГОСТ Р 34.10-2012 с длиной ключа 512 бит
szOID_DOMEN_ELIP_SIGN_ALG	1.2.643.2.2.19	Электронная подпись по ГОСТ Р 34.10-2001
szOID_CPCSP_ENCRYPT_ALG	1.2.643.2.2.21	Шифрование по ГОСТ 28147-89
szOID_CPCSP_IMITO_ALG	1.2.643.2.2.22	Алгоритм вычисления имитовставки. RFC 4357. ГОСТ 28147-89
szOID_CPCSP_PRF_ALG	1.2.643.2.2.23	Алгоритм выработки псевдослучайной последовательности (PRF) на основе ГОСТ Р 34.11-94
szOID_CPCSP_HASH_ALG	1.2.643.2.2.9	Хэширование по ГОСТ Р 34.11-94
szOID_CPCSP_HMAC_ALG	1.2.643.2.2.10	Алгоритм вычисления HMAC на основе ГОСТ Р 34.11-94. RFC 4490

Константа	Значение	Описание
szOID_CPCSP_HASH_SIGN_EL_ALG	1.2.643.2.2.3	Хэширование + электронная подпись по ГОСТ Р 34.10-2001
szOID_DH_EL_KEY_ALG	1.2.643.2.2.98	Протокол Диффи — Хеллмана по ГОСТ Р 34.10-2001
szOID_CPCSP_HASH_DEF_PARAM	1.2.643.2.2.30.1	Параметры хэширования по умолчанию
szOID_CPCSP_ENCR_DEF_PARAM	1.2.643.2.2.31.1	Параметры шифрования по умолчанию
szOID_CPCSP_ENCR_B_PARAM	1.2.643.2.2.31.2	Параметры шифрования B
szOID_CPCSP_ENCR_C_PARAM	1.2.643.2.2.31.3	Параметры шифрования C
szOID_CPCSP_ENCR_D_PARAM	1.2.643.2.2.31.4	Параметры шифрования D
szOID_CPCSP_EL_SIGN_DEF_PARAM	1.2.643.2.2.35.1	Параметры электронной подписи по умолчанию
szOID_CPCSP_EL_SIGN_1_PARAM	1.2.643.2.2.35.2	Набор параметров B для электронной подписи
szOID_CPCSP_EL_SIGN_2_PARAM	1.2.643.2.2.35.3	Набор параметров C для электронной подписи
szOID_CPCSP_EL_DH_DEF_PARAM	1.2.643.2.2.36.0	Набор параметров для DH по умолчанию
szOID_CPCSP_EL_DH_1_PARAM	1.2.643.2.2.35.1	Набор параметров 1 для DH
szOID_CSP2012_SIGN_256	1.2.643.7.1.1.1.1	Электронная подпись по ГОСТ Р 34.10-2012 с ключом 256 бит
szOID_CSP2012_SIGN_512	1.2.643.7.1.1.1.2	Электронная подпись по ГОСТ Р 34.10-2012 с ключом 512 бит
szOID_CSP2012_HASH_256	1.2.643.7.1.1.2.2	Хэширование по ГОСТ 34.11-2012 с длиной хэша 256 бит
szOID_CSP2012_HASH_512	1.2.643.7.1.1.2.3	Хэширование по ГОСТ 34.11-2012 с длиной хэша 512 бит
szOID_CSP2012_HASH_SIGN_256	1.2.643.7.1.1.3.2	Хэширование + электронная подпись по ГОСТ Р 34.10-2012 с ключом 256 бит
szOID_CSP2012_HASH_SIGN_512	1.2.643.7.1.1.3.3	Хэширование + электронная подпись по ГОСТ Р 34.10-2012 с ключом 512 бит
szOID_CSP2012_HMAC_256	1.2.643.7.1.1.4.1	HMAC по ГОСТ Р 34.11-2012 с ключом 256 бит

Константа	Значение	Описание
szOID_CSP2012_HMAC_512	1.2.643.7.1.1.4.2	HMAC по ГОСТ Р 34.11-2012 с ключом 512 бит
szOID_CSP2012_EXCHANGE_256	1.2.643.7.1.1.6.1	Согласование ключей по ГОСТ Р 34.10-2012 с ключом 256 бит
szOID_CSP2012_EXCHANGE_512	1.2.643.7.1.1.6.2	Согласование ключей по ГОСТ Р 34.10-2012 с ключом 512 бит
szOID_CSP2012_SIGN_512_PARAM_A	1.2.643.7.1.2.1.2.1	Набор параметров A для электронной подписи по ГОСТ 34.10-2012
szOID_CSP2012_ENCRYPT_PARAM	1.2.643.7.1.2.5.1.1	Набор параметров ISO/IEC 18033-3 для шифрования по ГОСТ 28147-89
szOID_CPCSP_TLS_PROXY	1.2.643.2.2.34.1	KeyUsage для прокси сертификата при построении TLS-соединения
szOID_REQUEST_VALIDITY	1.2.643.2.4.1.1.1.1.2	Идентификатор в запросе на сертификат желаемого срока действия сертификата

Идентификаторы алгоритмов

Таблица 4. Идентификаторы алгоритмов

Идентификатор	Значение (Hex)	Описание идентификатора
CPCSP_HASH_ID	801E	Идентификатор алгоритма хэширования по ГОСТ Р 34.11-94.
CPCSP_ENCRYPT_ID	661E	Идентификатор алгоритма шифрования по ГОСТ 28147 89.
CPCSP_IMITO_ID	801F	Идентификатор алгоритма имитозащиты по ГОСТ 28147-89.
CALG_PRO_EXPORT	661F	Идентификатор алгоритма защищенного экспорта ключа.
CALG_SIMPLE_EXPORT	6620	Идентификатор алгоритма простого экспорта ключа.
CALG_TLS1_MASTER_HASH	8020	Идентификатор алгоритма для выработки производного ключа в протоколе TLS1.
ELLIP_SIGN_ID	2E23	Идентификатор алгоритма электронной подписи по ГОСТ Р 34.10-2001.
CPCSP_DH_EL_ID	AA24	Идентификатор алгоритма обмена ключей по Диффи — Хеллману с использованием алгоритма ГОСТ Р 34.10-2001.
CALG_DH_EL_EPHEM	AA25	Идентификатор алгоритма создания эфемерных ключей для алгоритма обмена ключей по Диффи — Хеллману с использованием алгоритма ГОСТ Р 34.10-2001.
CALG_ITCS_EXPORT	662E	Идентификатор алгоритма экспорта/испорта ключа в формате ИнфоТеКС. Ключ экппортируется с дополнительной информацией, которая также защищена имитовставкой.
CALG_TLS1_MAC_KEY	6C03	Идентификатор алгоритма выработки ключа имитозащиты в протоколе TLS1.
CALG_TLS1_ENC_KEY	6C07	Идентификатор алгоритма выработки ключа шифрования в протоколе TLS1.
CALG_MASTER_DERIVE_HASH	8050	Идентификатор алгоритма выработки симметричного ключа из мастер-хэша.
CSP_HASH_2012_256BIT_ID	8021	Идентификатор алгоритма хэширования по ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит.

Идентификатор	Значение (Hex)	Описание идентификатора
CSP_HASH_2012_512BIT_ID	8022	Идентификатор алгоритма хэширования по ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит.
CSP_2012_256_SIGN_ID	2E49	Идентификатор алгоритма подписи по ГОСТ Р 34.10-2012 с ключом 256 бит.
CSP_2012_512_SIGN_ID	2E3D	Идентификатор алгоритма подписи по ГОСТ Р 34.10-2012 с ключом 512 бит.
CSP_2012_256_EXCHANGE_ID	AA46	Идентификатор алгоритма согласования по Диффи — Хеллману с использованием алгоритма ГОСТ Р 34.10-2012 с ключом 256 бит.
CSP_2012_512_EXCHANGE_ID	AA42	Идентификатор алгоритма согласования по Диффи — Хеллману с использованием алгоритма ГОСТ Р 34.10-2012 с ключом 512 бит.
CSP_2012_256_EXCHANGE_ID_EPHEM	AA47	Идентификатор алгоритма создания эфемерных ключей для алгоритма согласования по Диффи — Хеллману с использованием алгоритма ГОСТ Р 34.10-2012 с ключом 256 бит.
CSP_2012_512_EXCHANGE_ID_EPHEM	AA43	Идентификатор алгоритма создания эфемерных ключей для алгоритма согласования по Диффи — Хеллману с использованием алгоритма ГОСТ Р 34.10-2012 с ключом 512 бит.
CALG_PRO12_EXPORT	6621	CALG алгоритма экспорта/импорта ключа по Р 50.1.113-2016
CALG_TLS1PRF_2012_256	8031	Идентификатор алгоритма псевдослучайной функции (PRF) протокола по Р 50.1.113-2016 на основе функции хэширования с длиной выхода 256 бит
CALG_TLS1PRF_2012_512	8032	Идентификатор алгоритма псевдослучайной функции (PRF) протокола по Р 50.1.113-2016 на основе функции хэширования с длиной выхода 512 бит
CALG_TLS1_MASTER_HASH_2012_256	8036	Идентификатор алгоритма «ключевого мастера» по протоколу TLS на основе Рекомендаций по стандартизации ТК26 «Использование наборов алгоритмов шифрования на основе ГОСТ 28147-89 для протокола безопасности транспортного уровня» с длиной выхода 256 бит.
CALG_TLS1_MASTER	4C06	Идентификатор алгоритма исходного ключа для реализации протокола TLS1.

Таблица 5. Идентификаторы типов алгоритмов

Идентификатор	Значение (Hex)	Описание идентификатора
`ALG_TYPE_GR3410`	0E00	Тип алгоритма подписи.

Типы ключевых блобов при экспорте или импорте ключа

Таблица 6. Допустимые значения параметра bType в структурах и функциях экспорта и импорта ключей

Значение bType	Описание	Значение
`PUBLICKEYBLOB`	Предназначен для транспорта открытых ключей.	`0x6`
`PRIVATEKEYBLOB`	Предназначен для транспорта пар ключей (секретных ключей).	`0x7`
`SIMPLEBLOB`	Предназначен для транспорта сессионных ключей.	`0x1`
`OPAQUEKEYBLOB`	Предназначен для транспорта текущего состояния ключа.	`0x9`

Структуры криптопровайдера

Структура VTABLEPROVSTRUC

Структура предназначена для передачи информации о вызывающем приложении при открытии контекста криптопровайдера.

```
typedef struct _VTableProvStruc {
DWORD   Version;
FARPROC FuncVerifyImage;
FARPROC FuncReturnhWnd;
DWORD   dwProvType;
BYTE *pbContextInfo;
DWORD     cbContextInfo;
LPSTR pszProvName;
} VTableProvStruc, *PVTableProvStruc;
```

Подробнее о данной структуре см. соответствующий раздел MSDN https://msdn.microsoft.com/ru-ru/library/windows/desktop/aa388195(v=vs.85).aspx.

Структура PRIVATE_KEY_USAGE_PERIOD

Структура представляет собой расширение сертификата, указывающее период действия закрытого ключа, соответствующего сертификату.

```
typedef struct _PRIVATE_KEY_USAGE_PERIOD
{
    /// Дата и время начала действия закрытого ключа
    FILETIME notBefore;
    /// Дата и время окончания действия закрытого ключа
    FILETIME notAfter;
}
```

Подробнее см. RFC 3280 http://www.ietf.org/rfc/rfc3280, раздел 4.2.1.4 Private Key Usage Period.

Контейнеры ключей, имя контейнера

Под «контейнером ключей» в ViPNet CSP понимается именованный объект (файл, объект на устройстве PKCS#11), содержащий закрытый ключ, информацию о ключе (алгоритм, параметры, назначение ключа и другое), при необходимости открытый ключ и сертификат. Имя контейнера задается при его создании и передается в функцию открытия контекста криптопровайдера см. Открытие контекста провайдера (на стр. 22). Имя может быть полным, то есть содержать полный путь к файлу или объекту на устройстве. Допускается использование сокращенных имен. Перечень допустимых значений имен приведен в таблице ниже.

Таблица 7. Допустимые имена контейнеров

Значение	Описание
`NULL`	Используется только для временных контейнеров в памяти.
`Full Path`	Полный путь к файлу на диске, например, `C:\containers\cont1` — для ОС Windows. `/home/user1/.itcs/vipnet-csp/containers/` — для ОС Linux.
`Имя файла`	Имя контейнера находящегося по пути по умолчанию
`<Device String>`	Уникальный путь к объекту на устройстве. Представляется в специальном формате, может быть получен с использованием функции `CPGetProvParam` для открытого контейнера с параметром `PP_UNIQUE_CONTAINER`, см. Получение параметров провайдера (на стр. 25).
`\\!\<Тип Устройства>\<UID Устройства>\<имя Контейнера>`	При таком представлении имени осуществляется поиск контейнера с заданным именем на устройстве заданного типа и имеющим соответствующий идентификатор. Любой из элементов имени может быть опущен, при этом поиск контейнера будет осуществляться по оставшимся параметрам. Например, при передаче на вход функции открытия контекста строки вида: `\\!\ruToken\\cont1`. Будет открыт контейнер с именем `cont1` на устройстве ruToken. См. полный перечень допустимых типов устройств.
`\\.\<имя Считывателя>\<имя файла Контейнера>`	Формат FQCN (Fully Qualified Container Name), в качестве имени считывателя может быть указан любой, зарегистрированный в системе. Например: `\\.\ AKS ifdh 0\cont1`.

2

Инициализация и параметры криптопровайдера

Открытие контекста провайдера

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptAcquireContext(
    HCRYPTPROV * phProv,
    LPCSTR szContainer,
    LPCSTR szProvider,
    DWORD dwProvType,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPAcquireContext(
    HCRYPTPROV * phProv,
    CHAR * pszContainer,
    DWORD, dwFlags,
    PVTABLEPROVSTRUC pVTable
);
```

Параметры:

- `phProv [out]` — адрес, по которому функция копирует дескриптор криптопровайдера.
- `pszContainer [in]` — имя контейнера ключей. Это указатель на строку, длиной не больше, чем `MAX_PATH` знаков, включая признак конца строки. Описание синтаксиса и возможных значений имен приведено в разделе Контейнеры, имя контейнера (см. «Контейнеры ключей, имя контейнера» на стр. 20).
- `dwFlags [in]` — параметр имеет нулевое или одно из значений, приведенных в таблице ниже.

  Таблица 8. Значения параметра `dwFlags`

Значение dwFlags	Описание
`CRYPT_VERIFYCONTEXT`	Открытие контекста в памяти без доступа к секретным ключам. Используется для проверки подписи и хэширования данных.
`CRYPT_NEWKEYSET`	Создание нового контейнера с именем, соответствующим `pszContainer`. Если контейнер с данным именем уже существует, то вызов с данным флагом приведет к ошибке.
`CRYPT_DELETEKEYSET`	Контейнер ключей, соответствующий `pszContainer`, удаляется.
`CRYPT_SILENT`	Не использовать пользовательский интерфейс (UI) при выполнении операций с данным контекстом.

- `pVTable [in]` — указатель на структуру `VTABLEPROVSTRUC`. В зависимости от типа провайдера (`dwProvType`) будет использоваться провайдер для работы с парами ключей алгоритма ГОСТ Р

34.10-2001 (тип 2) или алгоритма ГОСТ Р 34.10-2012 (тип 77 и тип 78 для секретного ключа длины 256 и 512 бит соответственно).

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 9. Коды ошибок

Возвращаемые значения	Описание
`NTE_BAD_KEYSET`	Контейнер ключей не был открыт, поврежден или не существует.
`NTE_EXISTS`	Параметр `dwFlags` установлен в `CRYPT_NEWKEYSET`, а контейнер ключей уже существует.
`ERROR_INVALID_HANDLE`	Неверное значение адреса для копирования дескриптора провайдера.
`ERROR_DS_INSUFF_ACCESS_RIGHTS`	Нет прав на доступ к криптографическим функциям.
`ERROR_INVALID_PARAMETER`	Неверные параметры вызова.
`ERROR_CANCELLED`	Пользователь прервал операцию.
`NTE_KEYSET_NOT_DEF`	Контейнер ключей не может быть удален, так как не существует.
`NTE_SIGNATURE_FILE_BAD`	Нарушение целостности библиотеки криптопровайдера.
`NTE_PROVIDER_DLL_FAIL`	Ошибка инициализации библиотеки криптопровайдера.

Внимание! Контекст провайдера, открытый с помощью функции `CPAcquireContext`, должен быть закрыт с помощью функции `CPReleaseContext`. В противном случае:

- нарушаются требования по безопасности;
- могут возникнуть ошибки при закрытии программы, при этом информация о некорректном закрытии помещается в системный журнал Windows.

Закрытие контекста провайдера

Используется для удаления дескриптора криптопровайдера, созданного функцией `CPAcquireContext()`.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptReleaseContext(
    HCRYPTPROV hProv,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPReleaseContext (
    HCRYPTPROV hProv,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `dwFlags [in]` — значения флагов. Параметр зарезервирован для будущего использования и должен быть `0`.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 10. Коды ошибок

Возвращаемые значения	Описание
`NTE_BAD_FLAGS`	Параметр `dwFlags` имеет ненулевое значение.
`ERROR_INVALID_HANDLE`	Передан некорректный дескриптор контекста провайдера.

Получение параметров провайдера

Производит возвращение параметров криптопровайдера.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptGetProvParam(
    HCRYPTPROV hProv,
    DWORD dwParam,
    BYTE *pbData,
    DWORD *pdwDataLen,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPGetProvParam (
    HCRYPTPROV hProv,
    DWORD dwParam,
    BYTE * pbData,
    DWORD * pdwDataLen,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `dwParam [in]` — значение аргумента определяет тип запроса. В настоящее время определены следующие значения `dwParam`:

  Таблица 11. Значения параметра `dwParam`

Значение dwParam	Описание, содержимое буфера pbData
`PP_CONTAINER`	Имя контейнера ключей. Возвращает имя контейнера, заданное в параметре `pszContainer` функции открытия контейнера `CPAcquireContext`.
`PP_UNIQUE_CONTAINER`	Уникальное имя контейнера ключей. Возвращает полное, уникальное имя контейнера. В зависимости от носителя это либо полный путь к файлу, либо имя контейнера на устройстве Device String.
`PP_ENUMALGS`	Перебор списка поддерживаемых алгоритмов. `pbData` — указатель на структуру типа `PROV_ENUMALGS`.
`PP_ENUMALGS_EX`	Перебор списка поддерживаемых алгоритмов, `pbData` — указатель на структуру типа `PROV_ENUMALGS_EX`.

Значение dwParam	Описание, содержимое буфера pbData
PP_ENUMCONTAINERS	Перебор контейнеров ключей, обрабатываемых криптопровайдером. Чтобы выполнить перебор ключевых контейнеров, соответствующих компьютеру, вызовите функцию CryptAcquireContext, установив флаг CRYPT_MACHINE_KEYSET, и далее используйте значение, возвращаемое функцией CryptAcquireContext, в качестве параметра hProv при вызове функции CryptGetProvParam. По умолчанию из добавленных внешних контейнеров (контейнеры из cont_info.dat, то есть располагающиеся не в папках по умолчанию) перечисляются только те контейнеры, к которым имеется доступ в момент перечисления. Для перечисления всех внешних контейнеров необходимо использовать флаг CRYPT_NOSKIP в сочетании с данным параметром.
PP_PROVTYPE	Тип криптопровайдера. Задается величиной DWORD.
PP_NAME	Строковая величина с признаком конца строки, содержащая имя криптопровайдера.
PP_HASHOID	Идентификатор параметров хэширования (узел замены, подстановка) в контексте провайдера по умолчанию. См. ГОСТ Р 34.11-2001 и RFC 4357. В качестве pbData должна передаваться/возвращается строка, представляющая один из поддерживаемых идентификаторов параметров хэширования.
PP_SIGNATUREOID	Идентификатор параметров подписи (эллиптическая кривая) в контексте провайдера по умолчанию. См. ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012, RFC 4357, Р 50.1.114. В качестве pbData должен передаваться/возвращается строка, представляющая один из поддерживаемых идентификаторов параметров подписи.
PP_DHOID	Идентификатор параметров Диффи — Хелмана (эллиптическая кривая) в контексте провайдера по умолчанию. См. ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012, RFC 4357, Р 50.1.114. В качестве pbData должна передаваться/возвращается строка, представляющая один из поддерживаемых идентификаторов параметров Диффи — Хеллмана.
PP_ENUM_HASHOID	Перечисление поддерживаемых параметров хэширования (узел замены, подстановка). См. ГОСТ Р 34.11-2001 и RFC 4357. В качестве pbData возвращается строка, представляющая один из поддерживаемых идентификаторов параметров хэширования. Соглашения о вызовах при переборе значений аналогичны PP_ENUMCONTAINERS.

Значение dwParam	Описание, содержимое буфера pbData
PP_ENUM_CIPHEROID	Перечисление поддерживаемых параметров шифрования (узел замены, подстановка). См. ГОСТ 28147-89 и RFC 4357. В качестве pbData возвращается строка, представляющая один из поддерживаемых идентификаторов параметров шифрования. Соглашения о вызовах при переборе значений аналогичны PP_ENUMCONTAINERS.
PP_CIPHEROID	Идентификатор параметров шифрования (узел замены, подстановка) в контексте провайдера по умолчанию. См. ГОСТ 28147-89 и RFC 4357. В качестве pbData должна передаваться/возвращается строка, представляющая один из поддерживаемых идентификаторов параметров шифрования.
PP_ENUM_SIGNATUREOID	Перечисление поддерживаемых параметров подписи (эллиптическая кривая). См. ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012, RFC 4357, Р 50.1.114. В качестве pbData возвращается строка, представляющая один из поддерживаемых идентификаторов параметров подписи. Соглашения о вызовах при переборе значений аналогичны PP_ENUMCONTAINERS.
PP_ENUM_DHOID	Перечисление поддерживаемых параметров Диффи — Хелмана (эллиптическая кривая). См. ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012, RFC 4357, Р 50.1.114. В качестве pbData возвращается строка, представляющая один из поддерживаемых идентификаторов параметров Диффи — Хелмана. Соглашения о вызовах при переборе значений аналогичны PP_ENUMCONTAINERS.
PP_SHORT_CONTAINER_NAME	Возвращает строку, представляющую короткое имя контейнера.
PP_RNG_REGLAMENT_CTRL	Регламентный контроль ДСЧ. Возвращает результат запуска регламентного контроля ДСЧ. Параметр pbData не используется.
PP_VERSION	Версия криптопровайдера, возвращает 4 байта, старшие 2 соответствуют старшей, младшие — младшей составляющей. Для текущей версии 3.2 возвращается 0x00030002.
PP_KEYSPEC	Тип ключа (всегда либо AT_KEYEXCHANGE, либо AT_SIGNATURE, либо их комбинация). Возвращается в виде переменной типа DWORD.
PP_KEYSET_TYPE	Определяет, является ли hProv контекстом ключа компьютера или пользователя. Возвращает DWORD со значением CRYPT_MACHINE_KEYSET, если этот флаг был передан в CPAcquireContext(), или 0 в противном случае.

Значение dwParam	Описание, содержимое буфера pbData
PP_IMPTYPE	Тип реализации криптопровайдера. Возвращается в виде переменной типа DWORD. В настоящее время определены следующие типы реализации: • CRYPT_IMPL_HARDWARE; • CRYPT_IMPL_SOFTWARE; • CRYPT_IMPL_MIXED; • CRYPT_IMPL_UNKNOWN; • CRYPT_IMPL_REMOVABLE.
PP_SIG_KEYSIZE_INC	Шаг между максимальным и минимальным размером ключа подписи. Возвращает 0.
PP_KEYX_KEYSIZE_INC	Шаг между максимальным и минимальным размером ключа подписи. Возвращает 0.
PP_SMARTCARD_READER	Получение имени считывателя. Параметр pbData указывает на массив ANSI-символов, образованных строкой с завершающим нулем. Данная строка содержит имя считывателя для текущего контейнера. Размер буфера задается параметром pdwDataLen. **Примечание**. В ОС Windows Server 2003 этот параметр не поддерживается.

- pbData [out] — буфер данных параметра. Функция копирует соответствующие параметру данные в буфер. Формат этих данных зависит от значения dwParam. Если аргумент функции — NULL, то данные не копируются. Требуемый размер буфера в байтах возвращается в pdwDataLen.
- pdwDataLen [in/out] — если параметр PP_ENUMCONTAINERS задан, то первый вызов функции возвратит максимальный допустимый размер контейнера ключей, разрешенный текущим провайдером. В отличие от других возможных параметров, таких как возвращение длины самого длинного существующего контейнера или длины текущего контейнера. Последующий вызов функции перебора контейнеров ключей не изменит значение параметра dwLen. Для каждого перебранного контейнера вы можете программно определить длину строки по нулевому символу окончания строки, если это необходимо. Если перебираемое значение принято и параметр pbData равен NULL, то флаг CRYPT_FIRST должен быть задан для получения корректной длины.
- dwFlags [in] — значения флагов. В настоящее время определены следующие значения флагов:

Таблица 12. Значение параметра dwFlags

Значение dwFlags	Описание
CRYPT_FIRST	Когда читается параметр перечисления (например, PP_ENUMALGS, PP_ENUMALGS_EX или PP_ENUMCONTAINERS) и установлен этот флаг, должен быть возвращен первый элемент в списке перечисления. Иначе возвращается следующий элемент в списке.
CRYPT_UNIQUE	Перечисление уникальных имен контейнеров. Если данный флаг установлен, то при вызове функции CryptGetProvParam(PP_ENUMCONTAINERS) будет возвращаться уникальное имя контейнера, а не короткое.

При успешном завершении функция возвращает TRUE, в противном случае возвращается FALSE. Если возвращается FALSE, соответствующий код ошибки (см. таблицу) может быть получен через функцию GetLastError().

Таблица 13. Коды ошибок

Возвращаемые значения	Описание
ERROR_INVALID_HANDLE	Неверный дескриптор криптопровайдера.
ERROR_INVALID_PARAMETER	Неверные значения параметров.
ERROR_MORE_DATA	Размер буфера pbData не достаточен для копирования затребованных данных.
NTE_BAD_TYPE	dwParam определяет неизвестный параметр.
ERROR_NO_MORE_ITEMS	Список для перебора (контейнеров, алгоритмов и тому подобное) исчерпан.

Задание параметров провайдера

Производит установку параметров провайдера.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptSetProvParam(
    HCRYPTPROV hProv,
    DWORD dwParam,
    const BYTE *pbData,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPSetProvParam (
    HCRYPTPROV hProv,
    DWORD dwParam,
    BYTE * pbData,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `dwParam [in]` — аргумент, принимающий следующие значения:

Таблица 14. Значения параметра `dwParam`

Значение dwParam	Описание, содержимое буфера pbData
`PP_HASHOID`	Устанавливает идентификатор параметров алгоритма функции хэширования. По умолчанию используется значение `szOID_CPCSP_HASH_DEF_PARAM`.
`PP_CIPHEROID`	Устанавливает идентификатор параметров алгоритма шифрования. По умолчанию используется значение `szOID_CPCSP_ENCR_DEF_PARAM`.
`PP_SIGNATUREOID`	Устанавливает идентификатор параметров алгоритма подписи. По умолчанию используется значение `szOID_CPCSP_EL_SIGN_DEF_PARAM`.
`PP_DHOID`	Устанавливает идентификатор параметров алгоритма Диффи — Хеллмана. По умолчанию используется значение `szOID_CPCSP_EL_DH_DEF_PARAM`.
`PP_USER_ENCRYPTPARAMS`	Задает параметры шифрования (узел замены, подстановка), которые не поддерживаются провайдером. См. ГОСТ Р 28147-89, RFC 4357. В качестве `pbData` должна передаваться ASN.1-структура `Gost28147-89-ParamSetParameters` в соответствии с RFC 4357.

Значение dwParam	Описание, содержимое буфера pbData
PP_HASHOID	Устанавливает идентификатор параметров алгоритма функции хэширования. По умолчанию используется значение szOID_CPCSP_HASH_DEF_PARAM.
PP_RESET_SILENT_FLAG	Устанавливает «тихий режим». Параметр pbData игнорируется. После вызова функции CryptSetProvParam флаг CRYPT_SILENT, указанный при открытии провайдера, в контексте провайдера сбрасывается.
PP_DELETE_PIN_FROM_REGISTRY	Удаляет сохраненный ПИН-код. Параметр pbData игнорируется. После вызова функции CryptSetProvParam сохраненный ПИН-код для текущего открытого контейнера удаляется. Не поддерживается в Linux.
PP_DELETE_PASS_FROM_REGISTRY	Удаляет сохраненный пароль. Параметр pbData игнорируется. После вызова функции CryptSetProvParam сохраненный пароль для текущего открытого контейнера удаляется. Не поддерживается в Linux.
PP_KEYEXCHANGE_PIN	Задает пароль (PIN) для доступа к устройству, если данный контейнер находится на устройстве. Если контейнер находится на жестком диске, то задает пароль (PIN) для доступа к ключу, в противном случае он запрашивается у пользователя посредством UI (если не был установлен флаг CRYPT_SILENT. См. CPAcquireContext()). Если пароль задается для устройства, то могут быть использованы дополнительные флаги. Если вызов происходит с флагом PP_VERIFYPASS_FLAG, то пароль будет проверен на корректность. Вместе с PP_VERIFYPASS_FLAG может быть также использован флаг PP_SAVEPASS_TO_REGISTRY_FLAG, в случае чего проверенный пароль будет сохранен в реестр для устройства, ассоциированного с открытым контекстом провайдера.
PP_DELETE_PIN_FROM_REGISTRY	Удаляет PIN для устройства, ассоциированного с открытым контекстом провайдера.
PP_SIGNATURE_PIN	Идентично PP_KEYEXCHANGE_PIN.
PP_CHANGE_PASSWORD	Задает новый пароль для контейнера hProv.
PP_CLIENT_HWND	Дескриптор (handle) окна вызывающего приложения.
PP_USE_HARDWARE_RNG	Вызов биологического датчика случайных чисел («Электронной рулетки» (см. глоссарий, стр. 96)), если он еще не вызывался в данном сеансе работы. При наличии аппаратного датчика случайных числе инициализация производится при загрузке библиотеки, использование данного значения не эффективно.
PP_KEYSTORAGE	Специфический флаг для указания каталога с установленными ключами приложений ViPNet. Указатель pbData содержит строку — полный путь к каталогу, в который были установлены

Значение dwParam	Описание, содержимое буфера pbData
PP_HASHOID	Устанавливает идентификатор параметров алгоритма функции хэширования. По умолчанию используется значение szOID_CPCSP_HASH_DEF_PARAM. справочники и ключи приложений ViPNet.
PP_KEYSET_SEC_DESCR	Устанавливает права доступа на данный контейнер.
PP_CONTAINER_DEFAULT	Устанавливает данный контейнер как контейнер по умолчанию (контейнер с именем NULL).

- pbData [in] — буфер данных параметра. Этот буфер при обращении к функции должен содержать данные, которые соответствуют типу параметра, помещенному в dwParam. Формат данных зависит от типа параметра.
- dwFlags [in] — Значения флагов. Зарезервировано, должно быть 0.

При успешном завершении функция возвращает TRUE, в противном случае возвращается FALSE. Если возвращается FALSE, соответствующий код ошибки (см. таблицу) может быть получен через функцию GetLastError().

Таблица 15. Коды ошибок

Возвращаемые значения	Описание
ERROR_INVALID_HANDLE	Неверный дескриптор криптопровайдера.
ERROR_INVALID_PARAMETER	Неверные параметры вызова.
NTE_BAD_KEYSET	Нарушение целостности контейнера.
NTE_BAD_KEY	Нарушение целостности ключа в контейнере или неверный ключ защиты.
NTE_SILENT_CONTEXT	Только при задании PP_USE_HARDWARE_RNG. Вызов биологического датчика случайных чисел произведен для контекста, открытого в режиме CRYPT_SILENT.
NTE_BAD_FLAGS	Величина dwFlags имеет ненулевое значение.
NTE_BAD_DATA	Неверные данные.
NTE_BAD_TYPE	dwParam определяет неизвестный параметр.

3

Функции работы с ключами

Формирование ключа

Создание случайных ключей.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptGenKey(
    HCRYPTPROV hProv,
    ALG_ID Algid,
    DWORD dwFlags,
    HCRYPTKEY *phKey
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPGenKey (
    HCRYPTPROV hProv,
    ALG_ID AlgId,
    DWORD dwFlags,
    HCRYPTKEY * phKey
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `AlgId [in]` — идентификатор алгоритма шифрования или электронной подписи, для которого должен быть произведен ключ. Допустима передача любого из поддерживаемых алгоритмов (см. Идентификаторы алгоритмов (на стр. 15)). Кроме этого могут быть использованы следующие значения:

  Таблица 16. Значения параметра `AlgId`

AlgId	Описание
`AT_KEYEXCHANGE`	Производится пара ключей, сохраняемая в контейнере ключей. Алгоритм, используемый при выработке ключа по умолчанию, определяется типом провайдера. Алгоритм и параметры могут быть изменены при вызове функции `CPSetProvParam()`.
`AT_SIGNATURE`	Производится пара ключей, сохраняемая в контейнере ключей. Алгоритм, используемый при выработке ключа по умолчанию, определяется типом провайдера. Алгоритм и параметры могут быть изменены при вызове функции `CPSetProvParam()`.
`CALG_DH_EL_EPHEM`	Создание эфемерной пары ключей по алгоритму ГОСТ Р 34.10-2001.
`CSP_2012_256_EXCHANGE_ID_EPHEM`	Создание эфемерной пары ключей по алгоритму ГОСТ Р 34.10-2012 с длиной секретного ключа 256 бит.
`CSP_2012_512_EXCHANGE_ID_EPHEM`	Создание эфемерной пары ключей по алгоритму ГОСТ Р 34.10-2012 с длиной секретного ключа 512 бит.
`CALG_TLS1_MASTER`	Исходный ключ для реализации протокола TLS1.

- `dwFlags [in]` — флаги определяют признаки производимого ключа. Определены следующие флаги:

  Таблица 17. Значение параметра `dwFlags`

Значение dwFlags	Описание
`CRYPT_PREGEN`	Если этот флаг установлен, то генерируется «пустая» ключевая пара обмена. Параметры этой пары ключей должны быть установлены с использованием функции `CPSetKeyParam()`.

- `phKey [out]` — адрес, по которому функция копирует дескриптор сформированного ключа (пары открытый/закрытый ключи).

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 18. Коды ошибок

Возвращаемые значения	Описание
`ERROR_INVALID_HANDLE`	Неверный дескриптор криптопровайдера.
`NTE_BAD_KEYSET`	Нарушение целостности контейнера.
`NTE_BAD_KEY`	Неверный ключ (защиты контейнера).
`NTE_BAD_ALGID`	Параметр `AlgId` определяет алгоритм, не поддерживаемый криптопровайдером.
`NTE_NO_MEMORY`	Криптопровайдер во время операции исчерпал память.
`NTE_SILENT_CONTEXT`	Невозможно выполнение операции с установленным SILENT-режимом.
`ERROR_DS_INSUFF_ACCESS_RIGHTS`	Нет прав на доступ к криптографическим функциям.
`ERROR_INVALID_PARAMETER`	Неверные параметры вызова.
`NTE_PROVIDER_DLL_FAIL`	Ошибка внутреннего тестирования или инициализации датчика случайных чисел криптографической библиотеки.
`ERROR_CANCELLED`	Пользователь прервал операцию.

Получение доступа к ключу

Позволяет получить доступ к ключу в текущем контейнере.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptGetUserKey(
    HCRYPTPROV hProv,
    DWORD dwKeySpec,
    HCRYPTKEY *phUserKey
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPGetUserKey (
    HCRYPTPROV hProv,
    DWORD dwKeySpec,
    HCRYPTKEY * phUserKey
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `dwKeySpec [in]` — спецификация возвращаемого ключа. `ALG_ID` ключа, заданный при его создании или одна из спецификаций: `AT_KEYEXCHANGE` или `AT_SIGNATURE`. Функция возвращает первый из найденных ключей заданного типа. Перебор ключей в контейнере может быть реализован с использованием функции `CPGetProvParam()`.

  Таблица 19. Значения параметра `dwKeySpec`

dwKeySpec	Описание
`AT_KEYEXCHANGE`	Ключ шифрования.
`AT_SIGNATURE`	Ключ подписи.

- `phUserKey [out]` — адрес, по которому функция копирует дескриптор ключа.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 20. Коды ошибок

Возвращаемые значения	Описание
ERROR_INVALID_HANDLE	Неверный дескриптор криптопровайдера.
NTE_BAD_KEYSET	Нарушение целостности контейнера.
NTE_BAD_KEY	Неверный ключ.
ERROR_INVALID_PASSWORD	Неверный пароль или ключ защиты контейнера.
ERROR_DS_INSUFF_ACCESS_RIGHTS	Нет прав на доступ к криптографическим функциям.
NTE_NO_KEY	Ключ, указанный `dwKeySpec`-параметром, не существует.
NTE_NO_MEMORY	Провайдер во время работы исчерпал память.
NTE_BAD_ALGID	Неизвестный или неверный алгоритм или спецификация ключа.

Уничтожение дескриптора ключа

Освобождает дескриптор ключа, передаваемый через параметр `hKey`. После уничтожения дескриптора ключ не может использоваться.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptDestroyKey(HCRYPTKEY hKey)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPDestroyKey (
    HCRYPTPROV hProv,
    HCRYPTKEY hKey
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hKey [in]` — дескриптор уничтожаемого ключа.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 21. Код ошибки

Возвращаемые значения	Описания
`ERROR_INVALID_HANDLE`	Неверный дескриптор криптопровайдера.

Копирование дескриптора ключа

Создает копию заданного дескриптора ключа, включая все его переменные, определяющие внутреннее состояние ключа.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptDuplicateKey(
    HCRYPTKEY hKey,
    DWORD *pdwReserved,
    DWORD dwFlags,
    HCRYPTKEY *phKey
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPDuplicateKey (
    HCRYPTPROV hProv,
    HCRYPTKEY hKey,
    DWORD * pdwReserved,
    DWORD dwFlags,
    HCRYPTKEY * phKey
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hKey [in]` — дескриптор исходного (копируемого) ключа.
- `pdwReserved` — параметр зарезервирован для будущего использования и в настоящее время не используется.
- `dwFlags [in]` — параметр зарезервирован для будущего использования и должен быть `0`.
- `phKey [out]` — адрес, по которому функция возвращает дескриптор скопированного ключа.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 22. Коды ошибок

Возвращаемые значения	Описания
`ERROR_INVALID_HANDLE`	Неверный дескриптор провайдера или ключа.
`NTE_NO_MEMORY`	Криптопровайдер во время операции исчерпал память.

Задание параметров ключа

Устанавливает параметры ключа.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptSetKeyParam   (
    HCRYPTKEY hKey,
    DWORD dwParam,
    const BYTE *pbData,
    DWORD  dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPSetKeyParam (
    HCRYPTPROV hProv,
    HCRYPTKEY  hKey,
    DWORD      dwParam,
    BYTE  *    pbData,
    DWORD      dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hKey [in]` — дескриптор ключа, параметры которого устанавливаются.
- `dwParam [in]` — тип выставляемых параметров, возможные значения:

Таблица 23. Значения параметра `dwParam`

Значение dwParam	Содержимое буфера pbData
`KP_ALGID`	Идентификатор алгоритма ключа (`ALG_ID`), соответствующий данному ключу. Передается функции через буфер `pbData`.
`KP_EXPORTID`	Задание алгоритма экспорта для симметричного ключа. По умолчанию используется алгоритм, совместимый с криптопровайдером «КриптоПро», то есть `CALG_PRO_EXPORT` в соответствии с RFC4357, возможные дополнительные значения, используемые в приложениях ViPNet: `CALG_SIMPLE_EXPORT`, `CALG_ITCS_EXPORT`, `ALG_PRO12_EXPORT`.
`KP_IV`	Начальный вектор шифрования (IV). Последовательность байтов, содержащая IV, передается функции через буфер `pbData`.
`KP_SALT`	Зарезервировано в соответствии с описанием Microsoft CryptoAPI, в настоящий момент не используется.
`KP_MODE`	Режим шифрования, по умолчанию используется режим гаммирования с обратной связью `CRYPT_MODE_CFB`. Возможна установка значений для других режимов: `CRYPT_MODE_CBC`, `CRYPT_MODE_OFB`, `CRYPT_MODE_ECB`.

Значение dwParam	Содержимое буфера pbData
KP_MIXMODE	Дополнительный параметр ключа. Устанавливает режим преобразования ключа после зашифрования каждых 1024 байт информации. Режим преобразования ключа установлен по умолчанию. Выйти из этого режима можно, если для этого параметра pbData будет установлена в FALSE. Если pbData установлена в TRUE, устанавливается режим преобразования ключа. Размер pbData — 4 байта. Может быть установлен для ключей типа CALG_G28147. Задается величиной DWORD. **Внимание!** В режиме с отключенной модификацией ключа максимально доступный объем данных для шифрования на одном ключе — 4 Мбайт. При превышении данного объема функция шифрования вернет ошибку.
KP_OID	Идентификатор алгоритма. Применяется для определения параметров ключа, созданного с флагом CRYPT_PREGEN. Строковая величина с признаком конца строки.
KP_PADDING	Способ дополнения. Величина DWORD, содержащая метод дополнения, используемый шифром ключа, передается функции через буфер pbData. В настоящее время определены следующие способы дополнения: • DEFAULT_PADDING — дополнение по умолчанию; • PKCS5_PADDING — дополнение PKCS#5; • ZERO_PADDING — дополнение нулевыми байтами; • RANDOM_PADDING — дополнение случайными байтами; • UEC_PADDING — дополнение, принятое в карте УЭК; • NO_PADDING — дополнение отсутствует. Дополнение данных поддерживается только режимом шифрования CRYPT_MODE_CBC. При попытке выставления типа дополнения в других режимах будет возвращена ошибка.
KP_SIGNATUREOID	Идентификатор параметров ключа ГОСТ Р 34.10-2001 (для провайдера типа 2) ГОСТ Р 34.10-2012 (для провайдера типа 77 и 78), применяемых в алгоритме электронной подписи. Строковая величина с признаком конца строки.
KP_DHOID	Идентификатор параметров ключа ГОСТ Р 34.10-2001 (для провайдера типа 2) ГОСТ Р 34.10-2012 (для провайдера типа 77 и 78), применяемых в алгоритме Диффи — Хеллмана. Строковая величина с признаком конца строки.
KP_HASHOID	Идентификатор параметров алгоритма хэширования. Строковая величина с признаком конца строки.
KP_CIPHEROID	Идентификатор параметров алгоритма шифрования. Строковая величина с признаком конца строки.
KP_X	Секретный ключ в форме Little-endian. Данные передаются в форме BLOB-структуры, где член структуры pbData указывает на данные, а cbData передает длину данных. Применим только для пар ключей, сгенерированных с флагом CRYPT_PREGEN, при попытке вызова с другими ключами будет возвращена ошибка. Вызов с данным параметром и

Значение dwParam	Содержимое буфера pbData
	значением pbData=NULL инициирует генерацию пары ключей с заданными параметрами.
KP_CERTIFICATE	Сохранить сертификат в контейнере ключа.
KP_CLIENT_RANDOM	Установить значение случайной последовательности клиента (32 байта) для выработки общего ключа в протоколе TLS1.
KP_SERVER_RANDOM	Установить значение случайной последовательности сервера (32 байта) для выработки общего ключа в протоколе TLS1.
KP_SCHANNEL_ALG	Устанавливает структуру SCHANNEL_ALG, используемую при выработке ключей из мастер-ключа в протоколе TLS1. Передается как указатель на структуру SCHANNEL_ALG.
KP_PREHASH	Вызов процедуры генерации общего ключа для протокола TLS1. До вызова функции с эти параметром ключу должны быть выставлены параметры KP_CLIENT_RANDOM и KP_SERVER_RANDOM.

- pbData [in] — буфер данных параметра. При вызове функции буфер должен содержать данные, соответствующие значению параметра в dwParam. Формат данных зависит от типа параметра.
- dwFlags [in] — значения флагов. Параметр зарезервирован для будущего использования и должен быть 0.

При успешном завершении функция возвращает TRUE, в противном случае возвращается FALSE. Если возвращается FALSE, соответствующий код ошибки (см. таблицу) может быть получен через функцию GetLastError.

Таблица 24. Коды ошибок

Возвращаемые значения	Описание
NTE_BAD_FLAGS	Параметр dwFlags имеет ненулевое значение.
NTE_BAD_TYPE	Параметр dwParam передает неизвестное значение параметра.
NTE_BAD_KEY	Неверный ключ или нарушение целостности ключа.
ERROR_INVALID_HANDLE_STATE	Выполнение операции для ключа невозможно.
NTE_BAD_DATA	Неверные данные.
NTE_BAD_KEYSET	Контейнер поврежден.
ERROR_WRITE_FAULT	Ошибка записи на носитель.
NTE_BAD_ALGID	Неизвестный или неверный алгоритм ключа.

Получение параметров ключа

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptGetKeyParam(
    HCRYPTKEY hKey,
    DWORD dwParam,
    BYTE *pbData,
    DWORD *pdwDataLen,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPGetKeyParam (
    HCRYPTPROV hProv,
    HCRYPTKEY hKey,
    DWORD dwParam,
    BYTE * pbData,
    DWORD * pdwDataLen,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hKey [in]` — дескриптор ключа, параметры которого устанавливаются.
- `dwParam [in]` — тип получаемого параметра принимает следующие возможные значения:

Таблица 25. Значения параметра `dwParam`

Значение dwParam	Содержимое буфера pbData
`KP_ALGID`	Идентификатор алгоритма (`ALG_ID`), соответствующий данному ключу.
`KP_CONTAINER`	Полный путь к ключу, включающий имя субконтейнера в данном контейнере.
`KP_IV`	Начальный вектор инициализации (IV или синхропосылка) алгоритма шифрования.
`KP_SALT`	Зарезервировано в соответствии с описанием Microsoft CryptoAPI, в настоящий момент не используется.
`KP_MODE`	Режим шифрования, возможные значения режимов: `CRYPT_MODE_CBC`, `CRYPT_MODE_OFB`, `CRYPT_MODE_ECB`.
`KP_MIXMODE`	Дополнительный параметр ключа. Устанавливает режим преобразованием ключа после зашифрования каждых 1024 байт информации.
`KP_PADDING`	Способ дополнения. Величина DWORD, содержащая метод

Значение dwParam	Содержимое буфера pbData
	дополнения, используемый шифром ключа, передается функции через буфер `pbData`. В настоящее время определены следующие способы дополнения: • `DEFAULT_PADDING` — дополнение по умолчанию; • `PKCS5_PADDING` — дополнение PKCS#5; • `ZERO_PADDING` — дополнение нулевыми байтами; • `RANDOM_PADDING` — дополнение случайными байтами; • `UEC_PADDING` — дополнение, принятое в карте УЭК; • `NO_PADDING` — дополнение отсутствует. Дополнение данных поддерживается только режимом шифрования `CRYPT_MODE_CBC`. При попытке выставления типа дополнения в других режимах будет возвращена ошибка.
`KP_DHOID`	Идентификатор алгоритма Диффи — Хеллмана. Строковая величина с признаком конца строки.
`KP_SIGNATUREOID`	Идентификатор алгоритма подписи. Строковая величина с признаком конца строки.
`KP_CIPHEROID`	Идентификатор параметров алгоритма шифрования. Строковая величина с признаком конца строки.
`KP_Y`	Открытый ключ пары ключей.
`KP_CERTIFICATE`	Извлечь сертификат из контейнера ключа.
`KP_PERMISSIONS`	Флаг разрешений на использование ключа, указатель на DWORD. Возможные значения `CRYPT_EXPORT_KEY`, `CRYPT_IMPORT_KEY`, `CRYPT_ENCRYPT`, `CRYPT_DECRYPT`, `CRYPT_MAC`, `CRYPT_EXPORT`. Возможные комбинации флагов определяются типом ключа и флагами, заданными при его создании.
`KP_KEYLEN`	Длина ключа в битах.
`KP_BLOCKLEN`	Размер блока для блочного шифратора. В режиме CFB всегда возвращает `0`. Иные варианты в настоящий момент не поддерживаются.
`KP_HASHOID`	Вернуть идентификатор параметров алгоритма хэширования.
`KP_EXPIRATION_DATE`	Время окончания действия ключа. В качестве `pbData` возвращается параметр `FILETIME`, представляющий дату окончания срока действия ключа.

- `pbData [out]` — буфер данных параметра. Функция копирует соответствующие параметру данные в буфер. Формат этих данных зависит от значения `dwParam`. Если параметр — `NULL`, то данные не копируются. Требуемый размер буфера в байтах возвращается в `pdwDataLen`.
- `pdwDataLen [in/out]` — адрес длины данных параметра. При вызове функции указанный параметр содержит число байтов в буфере `pbData`. После ее выполнения параметр будет

установлен числом байтов данных параметра, скопированных в буфер `pbData`. Если буфер, соответствующий `pbData`, недостаточно велик, чтобы в него копировать запрошенные данные, через функцию `GetLastError()` будет возвращен код ошибки `ERROR_MORE_DATA`. В этом случае требуемый размер буфера возвращается в `pdwDataLen`. Если эта функция завершается с кодом ошибки, отличным от `ERROR_MORE_DATA`, в этом параметре возвращается ноль.

- `dwFlags [in]` — значения флагов. Параметр зарезервирован для будущего использования и должен быть `0`.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 26. Коды ошибок

Возвращаемые значения	Описание
`NTE_BAD_KEYSET`	Неверный контейнер ключа.
`NTE_BAD_KEY`	Неверный ключ.
`NTE_NOT_FOUND`	Объект не найден.
`ERROR_MORE_DATA`	Размер буфера `pbData` недостаточен для копирования затребованных данных.
`NTE_BAD_FLAGS`	Параметр `dwFlags` имеет ненулевое значение.
`NTE_BAD_TYPE`	Параметр `dwParam` передает неизвестное значение параметра.

Экспорт ключа из провайдера

Помещает данные ключа в ключевой блоб, предназначенный для внешнего хранения и передачи по каналам связи.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptExportKey(
    HCRYPTKEY hKey,
    HCRYPTKEY hExpKey,
    DWORD dwBlobType,
    DWORD dwFlags,
    BYTE *pbData,
    DWORD *pdwDataLen
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPExportKey (
    HCRYPTPROV hProv,
    HCRYPTKEY hKey,
    HCRYPTKEY hExpKey,
    DWORD dwBlobType,
    DWORD dwFlags,
    BYTE * pbData,
    DWORD * pdwDataLen
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hKey [in]` — дескриптор экспортируемого ключа.
- `hExpKey [in]` — дескриптор ключа, на котором осуществляется криптографическая защита экспортируемого ключа. Если ключевой блоб не должен быть зашифрован (например, тип ключевого блоба — `PUBLICKEYBLOB`) или для зашифрования ключа должен быть использован текущий ключ защиты, этот параметр должен быть нулевой. При экспорте ключевых блобов типов `PRIVATEKEYBLOB` и `SIMPLEBLOB` ключ должен быть зашифрован. Если ключ защиты не выставлен или не передан, функция возвращает ошибку. Параметры шифрования определяются для ключа защиты при его создании или с помощью функции `CPSetKeyParam`.
- `dwBlobType [in]` — тип ключевого блоба, предназначенного для экспорта ключа.

Таблица 27. Значения параметра `dwBlobType`

Значение dwBlobType	Описание
PRIVATEKEYBLOB	Секретный ключ в зашифрованном виде с дополнительной информацией.
PUBLICKEYBLOB	Открытый ключ пары ключей.
SIMPLEBLOB	Блоб, содержащий сессионный ключ шифрования, зашифрованный с использованием ключа экспорта.
OPAQUEKEYBLOB	Блоб, содержащий текущее состояние ключа.

- `dwFlags [in]` — значения флагов. Допустимое значение `NO_CHANGE_DH_EL_ID_FLAG`. Может использоваться для обеспечения совместимости режимов экспорта открытого ключа. По умолчанию параметр заголовка экспорта `aiKeyAlg` для ключей, созданных с типом `AT_KEYEXCHANGE`, заменяется на `ELLIP_SIGN_ID` (режим, совместимый с криптопровайдером «КриптоПРО»). При включении данного флага значение данного параметра будет равно `CPCSP_DH_EL_ID`.
- `pbData [in]` — буфер данных, куда функция копирует ключевой блоб.
- `pdwDataLen [in/out]` — адрес длины ключевого блоба. При вызове функции указанный параметр содержит число байтов в буфере `pbData`. После выполнения функции параметр будет установлен числом байтов данных, скопированных в буфер `pbData`. Если буфер, соответствующий `pbData`, недостаточно большой, будет возвращен код ошибки `ERROR_MORE_DATA` через функцию `SetLastError()`. В этом случае требуемый размер буфера возвращается в `pdwDataLen`. Если эта функция завершается с кодом ошибки, отличным от `ERROR_MORE_DATA`, в этом параметре возвращается ноль.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 28. Коды ошибок

Возвращаемые значения	Описание
ERROR_INVALID_PARAMETER	Неверные параметры вызова.
ERROR_INVALID_HANDLE	Неверный дескриптор криптопровайдера или ключа.
ERROR_NO_DATA	Нет ключей в контейнере.
ERROR_MORE_DATA	Буфер `pbData` недостаточно большой, чтобы копировать затребованные данные.
NTE_BAD_KEY	Один или оба из ключей, указанных `hKey` и `hExpKey`, недействительны.
NTE_BAD_KEYSET	Неверный контейнер.
NTE_BAD_ALGID	Неизвестный или неверный алгоритм ключа.

Возвращаемые значения	Описание
NTE_BAD_TYPE	dwBlobType параметр определяет неизвестный тип блоба.
NTE_BAD_KEYSET	Неверный дескриптор криптопровайдера.
ERROR_DS_INSUFF_ACCESS_RIGHTS	Нет прав на доступ к криптографическим функциям.

Импорт ключа в провайдер

Используется для импорта криптографического ключа из ключевого блоба в контейнер криптопровайдера.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptImportKey(
    HCRYPTPROV hProv,
    CONST BYTE *pbData,
    DWORD dwDataLen,
    HCRYPTKEY hPubKey,
    DWORD dwFlags,
    HCRYPTKEY *phKey
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPImportKey (
    HCRYPTPROV hProv,
    BYTE * pbData,
    DWORD dwDataLen,
    HCRYPTKEY hImpKey,
    DWORD dwFlags,
    HCRYPTKEY * phKey
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `pbData [in]` — буфер, содержащий ключевой блоб, содержащий открытый ключ, или секретный, экспортированный ранее с использованием функции `CPExportKey()` данным или другим криптопровайдером, функционирующим в том числе и на удаленном компьютере.
- `dwDataLen [in]` — длина ключевого блоба в байтах.
- `hImpKey [in]` — дескриптор ключа, на котором осуществляется снятие криптографической защиты импортируемого ключа. Значение этого параметра должно соответствовать значению `hExpKey`, определенному для функции `CPExportKey()` при экспорте ключевого блоба. Если ключевой блоб не зашифрован (например, `PUBLICKEYBLOB`), то этот параметр не используется и должен быть равен нулю. Параметр `hImpKey` может быть также выставлен в 0 при условии, что ключ экспорта был предварительно установлен в провайдер в качестве ключа защиты с использованием функции `CPSetProvParam()`.
- `dwFlags [in]` — значение флага. Если существующий в контейнере ключ должен быть заменен, то в параметр должен быть помещен флаг `CRYPT_UPDATE_KEY`, в противном случае ключ будет добавлен в контейнер.
- `phKey [out]` — адрес, по которому функция копирует дескриптор импортированного ключа.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError`.

Таблица 29. Коды ошибок

Возвращаемые значения	Описание
`ERROR_INVALID_PARAMETER`	Неверные параметры вызова.
`NTE_BAD_DATA`	Неверные данные или нарушение целостности ключевого блоба.
`NTE_BAD_KEY`	Один или оба из ключей, указанных `hKey` и `hImpKey`, недействительны.
`NTE_BAD_TYPE`	Тип ключевого блоба не поддерживается этим криптопровайдером и, возможно, ошибочен.
`ERROR_INVALID_HANDLE`	Ключ не найден.
`NTE_BAD_KEYSET`	Контейнер поврежден.
`NTE_BAD_ALGID`	Неверный или неизвестный тип алгоритма.
`ERROR_DS_INSUFF_ACCESS_RIGHTS`	Нет прав на доступ к криптографическим функциям.

Создание производного ключа

Создание ключа на основе данных пользователя, полученных в результате хэширования или присвоения значения с помощью функции `CPSetHashParam()` с параметром `HP_HASHVAL`.

Эта же функция позволяет получить криптографическую свертку двух ключей.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptDeriveKey(
    HCRYPTPROV hProv,
    ALG_ID Algid,
    HCRYPTHASH hBaseData,
    DWORD dwFlags,
    HCRYPTKEY *phKey
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPDeriveKey (
    HCRYPTPROV hProv,
    ALG_ID AlgId,
    HCRYPTHASH hBaseData,
    DWORD dwFlags,
    HCRYPTKEY * phKey
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `AlgId [in]` — идентификатор алгоритма шифрования, для которого должен быть произведен ключ. Значение `CPCSP_ENCRYPT_ID` параметра `AlgId` представляет собой ключ шифрования и/или имитозащиты данных по ГОСТ 28147-89. Впоследствии этот ключ можно пометить как ключ для импорта/экспорта с помощью функции `CPSetKeyParam()`.
- `hBaseData [in]` — дескриптор объекта функции хэширования, используемый для обработки входных данных.
- `dwFlags [in]` — флаги определяют признаки производимого ключа сессии. Флаг `CRYPT_UPDATE_KEY` обозначает следующее: если в провайдере имеется некоторый ключ, `hBaseData =NULL` и в качестве параметра `phKey` передан дескриптор другого ключа, то результатом выполнения функции будет криптографическая свертка двух ключей.
- `phKey [in/out]` — адрес, по которому функция копирует дескриптор произведенного ключа.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 30. Коды ошибок

Возвращаемые значения	Описание
ERROR_INVALID_HANDLE	Неверный дескриптор криптопровайдера или ключа.
NTE_NO_KEY	Ключ, необходимый для выполнения операции не найден.
NTE_BAD_ALGID	Параметр `AlgId` — определяет алгоритм, который не поддерживается криптопровайдером.
NTE_BAD_FLAGS	Величина `dwFlags` имеет ошибочное значение.
NTE_BAD_DATA	Неверные данные.
ERROR_INVALID_PARAMETER	Неверные значения параметров.
NTE_BAD_KEYSET	Контейнер поврежден.
NTE_BAD_KEY	Неверный ключ.

4

Датчик случайных чисел

Используется для получения случайной последовательности с датчиком случайных чисел.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptGenRandom(
    HCRYPTPROV hProv,
    DWORD dwLen,
    BYTE *pbBuffer
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPGenRandom (
    HCRYPTPROV hProv,
    DWORD dwLen,
    BYTE * pbBuffer
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `dwLen [in]` — число байтов случайных данных, которые будут произведены.
- `pbBuffer [in/out]` — буфер, куда копируются случайные данные. Длина этого буфера в байтах передается параметром `dwLen`.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 31. Коды ошибок

Возвращаемые значения	Описание
ERROR_INVALID_PARAMETER	Неверные значения параметров.
ERROR_INVALID_HANDLE	Неверный дескриптор криптопровайдера.
NTE_PROVIDER_DLL_FAIL	Ошибка внутреннего тестирования или инициализации датчика случайных чисел криптобиблиотеки.

5

Шифрование и расшифрование данных

Шифрование данных

Производит зашифрование данных. Одновременно может быть произведено вычисление имитозащитной вставки или хэширование данных.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptEncrypt(
    HCRYPTKEY hKey,
    HCRYPTHASH hHash,
    BOOL Final,
    DWORD dwFlags,
    BYTE *pbData,
    DWORD *pdwDataLen,
    DWORD dwBufLen
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPEncrypt (
    HCRYPTPROV hProv,
    HCRYPTKEY hKey,
    HCRYPTHASH hHash,
    BOOL Final,
    DWORD dwFlags,
    BYTE * pbData,
    DWORD * pdwDataLen,
    DWORD dwBufLen
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hKey [in]` — дескриптор ключа, используемого для зашифрования данных.
- `hHash [in]` — дескриптор объекта функции хэширования. Если значение хэш-функции не вычисляется, этот параметр равен нулю.
- `Final [in]` — булева величина. Определяет, является ли переданный функции блок последним зашифрованным блоком данных.
- `dwFlags [in]` — значения флагов. Параметр зарезервирован для будущего использования и должен быть 0.
- `pbData [in/out]` — буфер, содержащий данные для зашифрования. После зашифрования зашифрованные данные помещаются в этот же буфер.
- `pdwDataLen [in/out]` — адрес длины данных. Параметр `dwDataLen` определяет число байтов открытого текста в буфере `pbData`. Через этот параметр функция возвращает указатель на число байтов шифротекста, помещенного в буфер `pbData`.

- `dwBufLen [in]` — размер буфера `pbData` в байтах. Параметр определяет размер открытого текста, который, в силу использования дополнения до размера, кратного размеру блока при блочном шифровании, или по иным причинам, может не совпадать с размером шифротекста.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 32. Коды ошибок

Возвращаемые значения	Описание
`ERROR_INVALID_HANDLE`	Неверный дескриптор криптопровайдера или ключа.
`NTE_BAD_ALGID`	Ключ `hKey` определяет алгоритм, который данный криптопровайдер не поддерживает.
`NTE_BAD_FLAGS`	Величина `dwFlags` имеет ненулевое значение.
`NTE_BAD_KEY`	Ключ недействителен.
`NTE_BAD_LEN`	Размер буфера недостаточен для помещения в него шифротекста.
`ERROR_INVALID_PARAMETER`	Неверные значения параметров.
`ERROR_DS_INSUFF_ACCESS_RIGHTS`	Нет прав на доступ к криптографическим функциям.
`NTE_BAD_DATA`	Неверные данные.

При передаче ненулевого дескриптора объекта хэширования возможно возникновение дополнительных кодов ошибок из перечня кодов возврата функции `CPHashData()`.

Расшифрование данных

Производит расшифрование данных. Одновременно может быть произведено вычисление имитозащитной вставки или хэширование.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptDecrypt(
    HCRYPTKEY hKey,
    HCRYPTHASH hHash,
    BOOL Final,
    DWORD dwFlags,
    BYTE *pbData,
    DWORD *pdwDataLen
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPDecrypt (
    HCRYPTPROV hProv,
    HCRYPTKEY hKey,
    HCRYPTHASH hHash,
    BOOL Final,
    DWORD dwFlags,
    BYTE * pbData,
    DWORD * pdwDataLen
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hKey [in]` — дескриптор ключа, используемого для расшифрования данных.
- `hHash [in]` — дескриптор объекта хэширования. Если хэширование данных одновременно с их расшифрованием не требуется, этот параметр должен быть нулевой.
- `Final [in]` — булева величина. Определяет, является ли переданный функции блок последним расшифрованным блоком данных. Она должна быть установлена `TRUE`, если блок — последний или единственный, и `FALSE` — в противном случае.
- `dwFlags [in]` — значения флагов. Параметр зарезервирован для будущего использования и должен быть 0.
- `pbData [in/out]` — буфер, содержащий данные для расшифрования. После выполнения расшифрования открытый текст помещается в этот же буфер.
- `pdwDataLen [in/out]` — адрес длины данных. Параметр `dwDataLen` определяет число байтов шифротекста в буфере `pbData`. Через этот параметр функция возвращает указатель на число байтов открытого текста, помещенного в буфер `pbData`.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 33. Коды ошибок

Возвращаемые значения	Описание
`ERROR_INVALID_HANDLE`	Неверный дескриптор ключа или криптопровайдера.
`ERROR_INVALID_PARAMETER`	Неверные параметры вызова.
`NTE_BAD_ALGID`	Параметр `AlgId` определяет алгоритм, который не поддерживается криптопровайдером.
`NTE_BAD_KEY`	Неверный ключ.
`NTE_BAD_FLAGS`	Величина `dwFlags` имеет ненулевое значение.
`NTE_BAD_DATA`	Неверные данные.
`ERROR_DS_INSUFF_ACCESS_RIGHTS`	Нет прав на доступ к криптографическим функциям.

При передаче ненулевого дескриптора объекта хэширования возможно возникновение дополнительных кодов ошибок из перечня кодов возврата функции `CPHashData()`.

6

Хэширование и электронная подпись

Открытие контекста хэширования

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptCreateHash(
    HCRYPTPROV hProv,
    ALG_ID Algid,
    HCRYPTKEY hKey,
    DWORD dwFlags,
    HCRYPTHASH *phHash
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPCreateHash (
    HCRYPTPROV hProv,
    ALG_ID AlgId,
    HCRYPTKEY hKey,
    DWORD dwFlags,
    HCRYPTHASH * phHash
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `AlgId [in]` — идентификатор используемого алгоритма хэширования. В криптопровайдере определены следующие алгоритмы хэширования:
  - `CPCSP_HASH_ID` — алгоритм хэширования в соответствии с ГОСТ Р 34.11-94 с подстановкой «Верба-О»;
  - `CALG_TLS1_MASTER_HASH` — алгоритм выработки производных ключей для протокола TLS1.
  - `CALG_TLS1PRF` — алгоритм выработки псевдослучайной последовательности с использованием ключа для реализации протокола TLS1.
  - `CPCSP_IMITO_ID` — алгоритм имитозащиты в соответствии с ГОСТ 28147-89 с подстановкой «Верба-О»;
  - `CSP_HASH_2012_256BIT_ID` — алгоритм хэширования по ГОСТ Р 34.11-2012 с длиной хэш-кода 256 бит;
  - `CSP_HASH_2012_512BIT_ID` — алгоритм хэширования по ГОСТ Р 34.11-2012 с длиной хэш-кода 512 бит.
- `hKey [in]` — если используется алгоритм имитозащиты по ГОСТ 28147-89 (например, `CPCSP_IMITO_ID`), в этом параметре передается ключ, на котором осуществляется имитозащита. Для алгоритмов хэширования (таких как ГОСТ Р 34.11-94) значение параметра должно быть равно нулю.
- `dwFlags [in]` — значения флагов: `LONG_IMIT` — вернуть имитовставку длиной 8 байт.

- `phHash [out]` — адрес, по которому функция копирует дескриптор открытого контекста хэширования.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 34. Коды ошибок

Возвращаемые значения	Описание
`ERROR_INVALID_HANDLE`	Неверный дескриптор криптопровайдера или ключа.
`ERROR_INVALID_PARAMETER`	Неверные параметры вызова.
`NTE_BAD_KEY`	Неверный ключ.
`NTE_BAD_ALGID`	Параметр `AlgId` определяет алгоритм, который не поддерживается криптопровайдером.
`NTE_BAD_FLAGS`	Величина `dwFlags` имеет ненулевое значение.
`NTE_NO_MEMORY`	Криптопровайдер во время операции исчерпал память.

Закрытие контекста хэширования

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptDestroyHash(HCRYPTHASH hHash)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPDestroyHash(
    HCRYPTPROV hProv,
    HCRYPTHASH hHash
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hHash [in]` — дескриптор освобождаемого контекста хэширования.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`.

Таблица 35. Код ошибки

Возвращаемые значения	Описание
`ERROR_INVALID_HANDLE`	Неверный дескриптор криптопровайдера.

Копирование контекста хэширования

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptDuplicateHash(
    HCRYPTHASH hHash,
    DWORD *pdwReserved,
    DWORD dwFlags,
    HCRYPTHASH *phHash
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPDuplicateHash(
    HCRYPTPROV hProv,
    HCRYPTHASH hHash,
    DWORD * pdwReserved,
    DWORD dwFlags,
    HCRYPTHASH * phHash
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hHash [in]` — дескриптор контекста хэширования, создается через запрос функции `CPCreateHash()`.
- `pdwReserved [in]` — параметр зарезервирован для будущего использования и должен быть `NULL`.
- `dwFlags [in]` — параметр зарезервирован для будущего использования и должен быть 0.
- `phHash [out]` — адрес, по которому функция копирует дескриптор нового контекста хэширования.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 36. Коды ошибок

Возвращаемые значения	Описание
`NTE_NO_MEMORY`	Криптопровайдер во время операции исчерпал память.
`ERROR_INVALID_HANDLE`	Неверный дескриптор криптопровайдера.
`NTE_BAD_KEY`	Неверный ключ.
`ERROR_INVALID_PARAMETER`	Неверные параметры.

Задание параметров контекста хэширования

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptSetHashParam(
    HCRYPTHASH hHash,
    DWORD dwParam,
    const BYTE *pbData,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPSetHashParam (
    HCRYPTPROV hProv,
    HCRYPTHASH hHash,
    DWORD dwParam,
    BYTE * pbData,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hHash [in]` — дескриптор контекста хэширования, параметры которого устанавливаются.
- `dwParam [in]` — параметр, принимающий следующие возможные значения:

Таблица 37. Значения параметра `dwParam`

Значение dwParam	Содержимое буфера pbData
HP_HASHVAL	Устанавливает текущее значение хэш-функции.
HP_TLS1PRF_SEED	Задание параметра SEED для выработки псевдослучайной последовательности в алгоритме CALG_TLS1PRF.
HP_TLS1PRF_LABEL	Задание параметра LABLE для выработки псевдослучайной последовательности в алгоритме CALG_TLS1PRF.
HP_OPAQUEBLOB	Устанавливает состояние контекста хэширования (то есть предполагается, что это состояние было получено с помощью функции CPGetHashParam с одноименным идентификатором).
HP_HASHSTARTVECT	Вычисляет имитовставку. В качестве pbData должна передаваться последовательность байтов, представляющих начальное значение. Может использоваться только до начала хэширования данных.
HP_TLS1PRF_LABEL	Label для функции PRF в соответствии с RFC 5246, RFC 4357 и P 50.1.113. Применим к функции CryptSetHashParam для подсчета PRF. В качестве pbData должна передаваться последовательность байт, представляющих lebel. Не поддерживается в Linux.

- `pbData [in]` — буфер данных параметра. При вызове функции буфер должен содержать данные, соответствующие значению параметра в `dwParam`. Формат данных зависит от типа параметра `dwParam`.
- `dwFlags [in]` — значения флагов. Параметр зарезервирован для будущего использования и должен быть 0.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError`().

Таблица 38. Коды ошибок

Возвращаемые значения	Описание
NTE_BAD_FLAGS	Параметр dwFlags имеет ненулевое значение.
ERROR_INVALID_HANDLE	Неверный дескриптор криптопровайдера или объекта хэширования.
NTE_BAD_HASH	Неверный объект хэширования.
ERROR_INVALID_PARAMETER	Параметр dwParam передает неизвестное значение параметра.

Получение параметров контекста хэширования

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptGetHashParam(
    HCRYPTHASH hHash,
    DWORD dwParam,
    BYTE *pbData,
    DWORD *pdwDataLen,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPGetHashParam (
    HCRYPTPROV hProv,
    HCRYPTHASH hHash,
    DWORD dwParam,
    BYTE * pbData,
    DWORD * pdwDataLen,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hHash [in]` — дескриптор контекста хэширования, параметры которого извлекаются.
- `dwParam [in]` — параметр, принимающий следующие возможные значения:

Таблица 39. Значения параметра `dwParam`

Значение dwParam	Содержимое буфера pbData
`HP_ALGID`	Алгоритм функции хэширования. Идентификатор алгоритма (`ALG_ID`), соответствующий данному контексту, возвращается через буфер `pbData`. В `pdwDataLen` будет возвращена величина 4.
`HP_HASHSIZE`	Размер значения функции хэширования. Величина DWORD, определяющая число байтов в значении функции хэширования, будет возвращена через буфер `pbData`. В `pdwDataLen` будет возвращена величина 4.
`HP_HASHVAL`	Значение функции хэширования. Вычисленное значение функции хэширования будет возвращено через буфер `pbData`. Длина значения функции хэширования будет возвращена в `pdwDataLen`.

Значение dwParam	Содержимое буфера pbData
HP_OPAQUEBLOB	Возвращает полное состояние контекста хэширования (вызывается для восстановления его при помощи вызова CPGetHashParam с одноименным параметром).

- pbData [out] — буфер данных параметра. Функция копирует соответствующие параметру данные в буфер. Формат этих данных зависит от значения dwParam. Если параметр — NULL, то данные не копируются. Требуемый размер буфера в байтах возвращается в pdwDataLen.
- pdwDataLen [in/out] — адрес длины данных параметра. При вызове функции указанный параметр содержит число байтов в буфере pbData. После ее исполнения параметр будет установлен числом байтов данных параметра, скопированных в буфер pbData. Если буфер, соответствующий pbData недостаточно большой, чтобы в него копировать запрошенные данные, будет возвращен код ошибки ERROR_MORE_DATA через функцию SetLastError(). В этом случае требуемый размер буфера возвращается в pdwDataLen. Если функция завершается с кодом ошибки, отличным от ERROR_MORE_DATA, в этом параметре возвращается ноль.
- dwFlags [in] — значения флагов. Параметр зарезервирован для будущего использования и должен быть 0.

При успешном завершении функция возвращает TRUE, в противном случае возвращается FALSE. Если возвращается FALSE, соответствующий код ошибки (см. таблицу) может быть получен через функцию GetLastError().

Таблица 40. Коды ошибок

Возвращаемые значения	Описание
ERROR_INVALID_HANDLE	Неверный дескриптор провайдера или объекта хэширования.
ERROR_MORE_DATA	Размер буфера pbData недостаточен для копирования затребованных данных.
NTE_BAD_FLAGS	Параметр dwFlags имеет ненулевое значение.
ERROR_INVALID_PARAMETER	Параметр dwParam передает неизвестное значение параметра.
NTE_BAD_HASH	Неверный объект хэширования.
NTE_BAD_ALGID	Неверный алгоритм.
NTE_NO_MEMORY	Провайдер во время работы исчерпал память.

Хэширование данных

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptHashData(
    HCRYPTHASH hHash,
    const BYTE *pbData,
    DWORD dwDataLen,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPHashData (
    HCRYPTPROV hProv,
    HCRYPTHASH hHash,
    BYTE * pbData,
    DWORD dwDataLen,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hHash [in]` — дескриптор контекста хэширования. Приложение получает этот дескриптор, используя функцию `CPCreateHash()`.
- `pbData [in]` — адрес хэшируемых данных.
- `dwDataLen [in]` — число байтов хэшируемых данных.
- `dwFlags [in]` — значения флагов. Параметр зарезервирован для будущего использования и должен быть 0.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 41. Коды ошибок

Возвращаемые значения	Описание
`ERROR_INVALID_HANDLE`	Неверный дескриптор криптопровайдера или объекта хэширования.
`NTE_BAD_ALGID`	Ключ `hKey` определяет алгоритм, который данный криптопровайдер не поддерживает.
`NTE_BAD_HASH_STATE`	Была сделана попытка добавить данные к объекту функции хэширования, который уже отмечен как «закрытый».
`NTE_BAD_HASH`	Неверный объект хэширования.

Возвращаемые значения	Описание
NTE_BAD_KEY	Неверный ключ.
NTE_BAD_FLAGS	Данный код ошибки возвращается, если заданы флаги.

Хэширование сессионного ключа

Функция `CPHashSessionKey` передает криптографический ключ указанному объекту функции хэширования. Это позволяет хэшировать ключи сессии без передачи ключа приложению.

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptHashSessionKey(
    HCRYPTHASH  hHash,
    HCRYPTKEY   hKey,
    DWORD       dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPHashSessionKey (
    HCRYPTPROV hProv,
    HCRYPTHASH hHash,
    HCRYPTKEY hKey,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hHash [in]` — дескриптор объекта функции хэширования. Приложение получает этот дескриптор, используя функцию `CPCreateHash()`.
- `hKey [in]` — дескриптор хэшируемого ключа сессии.
- `dwFlags [in]` — значения флагов. Параметр зарезервирован. До будущего использования устанавливается в `NULL`.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError`.

Таблица 42. Коды ошибок

Возвращаемые значения	Описание
ERROR_INVALID_HANDLE	Неверный дескриптор провайдера или объекта хэширования.
NTE_BAD_ALGID	Параметр AlgId определяет алгоритм, который не поддерживается криптопровайдером.
NTE_BAD_FLAGS	Величина dwFlags имеет ненулевое значение.
NTE_BAD_HASH_STATE	Была сделана попытка добавить данные к объекту функции хэширования, который уже отмечен как «закрытый».
NTE_BAD_HASH	Неверный объект хэширования.
NTE_BAD_KEY	Тип хэшируемого ключа не соответствует ALG_TYPE_BLOCK.

Формирование электронной подписи

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptSignHash(
    HCRYPTHASH hHash,
    DWORD dwKeySpec,
    LPCWSTR szDescription,
    DWORD dwFlags,
    BYTE *pbSignature,
    DWORD *pdwSigLen
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPSignHash (
    HCRYPTPROV hProv,
    HCRYPTHASH hHash,
    DWORD dwKeySpec,
    LPCWSTR sDescription,
    DWORD dwFlags,
    BYTE * pbSignature,
    DWORD * pdwSigLen
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hHash [in]` — дескриптор контекста хэширования, который должен быть подписан.
- `dwKeySpec [in]` — тип пары ключей, использующейся при подписи значения функции хэширования. Может быть использован идентификатор алгоритма ключа (ALG_ID) или спецификация ключа:

  Таблица 43. Значения параметра `dwKeySpec`

Значение	Описание
`AT_KEYEXCHANGE`	Пара ключей обмена.
`AT_SIGNATURE`	Пара ключей цифровой подписи.

- `sDescription [in]` — в данной версии не поддерживается, должен быть установлен в `NULL`.
- `dwFlags [in]` — значения флагов. Параметр зарезервирован для будущего использования и должен быть 0.

- `pbSignature [in]` — буфер, через который возвращается значение подписи. Если через этот параметр передается NULL, то подпись не вычисляется. В этом случае требуемый размер буфера (в байтах) возвращается через параметр `pdwSigLen`.
- `pdwSigLen [in/out]` — адрес длины данных подписи. При вызове функции указанный параметр должен содержать число байтов в буфере `pbSignature`. После ее исполнения параметр будет установлен числом байтов данных, скопированных в буфер `pbSignature`. Если буфер, соответствующий `pbSignature`, недостаточно большой, чтобы в него копировать подпись, будет возвращен код ошибки `ERROR_MORE_DATA` через функцию `SetLastError()`. В этом случае требуемый размер буфера возвращается в `pdwSigLen`. Если эта функция завершается с кодом ошибки, отличным от `ERROR_MORE_DATA`, в этом параметре возвращается ноль.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError()`.

Таблица 44. Коды ошибок

Возвращаемые значения	Описание
NTE_BAD_FLAGS	`dwFlags` параметр отличен от нуля, либо параметр `dwKeySpec` содержит ошибочную величину.
NTE_BAD_KEY	Секретный ключ, указанный `dwKeySpec`, не найден.
ERROR_MORE_DATA	`pbSignature` буфер мал для копирования затребованных данных.
ERROR_CANCELLED	Пользователь прервал операцию.
ERROR_INVALID_HANDLE	Неверный дескриптор провайдера или объекта хэширования.
ERROR_DS_INSUFF_ACCESS_RIGHTS	Ошибка доступа к криптографическим функциям.
ERROR_INVALID_PARAMETER	Неверные параметры вызова.
NTE_NO_KEY	Ключ не найден.
NTE_BAD_KEYSET	Нарушение целостности контейнера.
NTE_SILENT_CONTEXT	Операция не может быть выполнена для контейнера, открытого с флагом `CRYPT_SILENT`.
ERROR_INVALID_PASSWORD	Неверный пароль или ключ защиты.
NTE_BAD_HASH	Неверный объект хэширования.

Примечание. Перед вызовом функции `CPSignHash()` должен быть получен дескриптор объекта хэширования через вызов функции `CPCreateHash()`. После этого используется функция `CPHashData()`, чтобы поместить данные в объект хэширования.

Функция `CPSignHash()` выполняет следующие внутренние шаги:

- объект функции хэширования «закрывается»; извлекается значение функции хэширования;
- значение функции хэширования дополняется, как это требуется алгоритмом подписи;
- вычисляется фактическое значение подписи.

После того как объект функции хэширования подписан, приложение не может добавлять к нему новые данные. Приложение должно разрушить объект функции хэширования через вызов функции `CPDestroyHash()`.

Проверка электронной подписи

Прототип функции, применимый в ОС Windows и Linux (рекомендуется к использованию):

```
BOOL WINAPI CryptVerifySignature(
    HCRYPTHASH hHash,
    const BYTE *pbSignature,
    DWORD dwSigLen,
    HCRYPTKEY hPubKey,
    LPCWSTR szDescription,
    DWORD dwFlags
)
```

Прототип функции, применимый только в ОС Windows:

```
BOOL WINAPI CPVerifySignature (
    HCRYPTPROV hProv,
    HCRYPTHASH hHash,
    BYTE * pbSignature,
    DWORD dwSigLen,
    HCRYPTKEY hPubKey,
    LPCWSTR sDescription,
    DWORD dwFlags
);
```

Параметры:

- `hProv [in]` — дескриптор криптопровайдера. Указанный дескриптор получается через запрос функции `CPAcquireContext()`.
- `hHash [in]` — дескриптор контекста хэширования, подпись которого проверяется.
- `pbSignature [in]` — буфер, содержащий значение проверяемой подписи.
- `dwSigLen [in]` — длина (в байтах) значения подписи.
- `hPubKey [in]` — открытый ключ подписи.
- `sDescription [in]` — в данной версии не поддерживается, должен быть установлен в `NULL`.
- `dwFlags [in]` — значения флагов. Параметр зарезервирован для будущего использования и должен быть `0`.

При успешном завершении функция возвращает `TRUE`, в противном случае возвращается `FALSE`. Если возвращается `FALSE`, соответствующий код ошибки (см. таблицу) может быть получен через функцию `GetLastError`.

Таблица 45. Коды ошибок

Возвращаемые значения	Описание
`ERROR_INVALID_HANDLE`	Неверный дескриптор провайдера или объекта хэширования.
`NTE_BAD_FLAGS`	`dwFlags` параметр отличен от нуля.
`NTE_BAD_ALGID`	Дескриптор `hHash` или `hPubKey` определяют алгоритм, который этот криптопровайдер не поддерживает.
`NTE_BAD_KEY`	Параметр `hPubKey` содержит дескриптор недопустимого открытого ключа.
`NTE_BAD_SIGNATURE`	Подпись неверна.

Примечание. Функция `CPVerifySignature` выполняет следующие внутренние шаги:

- объект функции хэширования «закрывается» и значение функции хэширования извлекается;
- значение функции хэширования дополняется, как это требуется алгоритмом подписи;
- осуществляется проверка подписи с использованием открытого ключа `hPubKey`;
- если подпись, переданная через буфер `pbSignature`, и вычисленное значение подписи не совпадают, функция возвращает `FALSE` с кодом ошибки `NTE_BAD_SIGNATURE`.

После выполнения проверки подписи приложение не может добавлять новые данные к объекту функции хэширования. Приложение должно разрушить объект функции хэширования через вызов функции `CPDestroyHash()`.

Формирование расширенной электронной подписи

Общие сведения о стандарте CAdES-BES

Криптопровайдер ViPNet CSP поддерживает формирование расширенной электронной подписи формата CAdES-BES.

CAdES-BES (CMS Advanced Electronic Signatures – Basic Electronic Signature) — основной и простейший формат электронной подписи, описываемый в стандарте CAdES. Он обеспечивает базовую проверку подлинности данных и защиту их целостности.

Электронная подпись, выполненная в формате CAdES-BES, содержит следующие атрибуты:

- подписываемые данные пользователя (определено в CMS и ESS, RFC 3852 (http://www.ietf.org/rfc/rfc3852), RFC 2634 (http://www.ietf.org/rfc/rfc2634) и RFC 5035 (http://tools.ietf.org/html/rfc5035); под подписываемыми данными понимается документ или сообщение подписывающей стороны);
- набор обязательных подписываемых атрибутов (определено в стандарте CAdES; атрибуты называются подписанными, если генерация подписи происходит от совокупности этих атрибутов и данных пользователя);
- значение цифровой подписи, вычисленное для данных пользователя и подписываемых атрибутов (определено в CMS, RFC 3852 (http://www.ietf.org/rfc/rfc3852)).

Обязательными подписываемыми атрибутами CAdES-BES являются:

- `content-type` (определен в CMS, RFC 3852 (http://www.ietf.org/rfc/rfc3852));
- `message-digest` (определен в CMS, RFC 3852 (http://www.ietf.org/rfc/rfc3852));
- `ESS signing-certificate` (определен в ESS, RFC 2634 (http://www.ietf.org/rfc/rfc2634)) или `ESS signing-certificate-v2` (определен в ESS update, RFC 5035 (http://tools.ietf.org/html/rfc5035)).

Реализация формирования электронной подписи в формате CAdES-BES

Если на вашем компьютере установлен криптопровайдер ViPNet CSP, любые подписанные CMS-сообщения, созданные с помощью интерфейса Microsoft CryptoAPI, соответствуют формату CAdES-BES. Исключение составляют следующие случаи:

- Использование CAdES-BES отключено (см. «Отключение выставления атрибутов CAdES-BES» на стр. 79).
- Сертификаты подписантов не найдены (см. «Действия в случае, если соответствующие сертификаты не найдены» на стр. 80).

Внимание! Обязательные подписываемые атрибуты, делающие CMS-сообщение удовлетворяющим стандарту CAdES (формат CAdES-BES), добавляются как при формировании электронной подписи по одному из поддерживаемых алгоритмов ГОСТ, так и при формировании электронной подписи по любому другому поддерживаемому алгоритму.

Отключение выставления атрибутов CAdES-BES

При необходимости вы можете отключить использование расширенной электронной подписи следующими способами:

- В функциях `CryptMsgOpenToEncode` и `CryptMsgControl` выставьте флаги `CMSG_CADES_DISABLE(=0x00000200)`.
- С помощью реестра ОС Windows. Для этого выполните следующие действия:
  - Если вы используете 32-разрядную ОС Windows, перейдите в раздел реестра `HKEY_LOCAL_MACHINE\SOFTWARE\InfoTeCS\CSP`.
  - Если вы используете 64-разрядную ОС Windows, перейдите в раздел реестра `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\InfoTeCS\CSP`.

Примечание. К этому разделу реестра обращаются как 32-разрядные, так и 64-разрядные приложения.

  - Создайте параметр DWORD с именем `AddCadesBesAttribute` и значением `0`.

Действия в случае, если соответствующие сертификаты не найдены

При формировании электронной подписи в формате CAdES-BES требуется не только закрытый ключ, но и соответствующий ему сертификат. Сертификат необходим для выбора обязательного подписываемого атрибута CAdES_BES и для вычисления выбранного атрибута.

Если при формировании CMS-сообщения соответствующие сертификаты не были найдены, выполняются следующие действия:

- Если в функциях `CryptMsgOpenToEncode` и `CryptMsgControl` указан флаг `CMSG_CADES_STRICT (=0x0000010)`, формирование подписи прекращается и выдается ошибка.
- Если в функциях `CryptMsgOpenToEncode` и `CryptMsgControl` флаг `CMSG_CADES_STRICT` не указан, электронная подпись будет сформирована, однако в этом случае она не будет содержать атрибут `ESS signing-certificate` (`ESS signing-certificate-v2`) и поэтому не будет соответствовать формату CAdES-BES.

Работа с обязательными подписываемыми атрибутами CAdES-BES

При формировании электронной подписи в формате CAdES-BES требуется не только закрытый ключ, но и соответствующий ему сертификат. Сертификат необходим для выбора обязательного подписываемого атрибута CAdES_BES (`ESS signing-certificate` или `ESS signing-certificate-v2`), а также для вычисления выбранного атрибута.

Поиск сертификата, соответствующего закрытому ключу, с помощью которого формируется электронная подпись, выполняется по идентификатору `SignerIdentifier`. Этот идентификатор находится в CMS-сообщении в структуре `SignerInfo` (http://www.ietf.org/rfc/rfc3852).

Примечание. В структуре `SignerInfo` CMS-сообщения также находится сама электронная подпись.

Поиск сертификата, соответствующего закрытому ключу, с помощью которого формируется электронная подпись, выполняется последовательно в следующих местоположениях:

1. Список сертификатов из CMS-сообщения (см. поле `Certificates` структуры `SignedData` в RFC 3852 (http://www.ietf.org/rfc/rfc3852)).
2. Контейнер ключей.
3. Системное хранилище сертификатов **Личное** (**My**) текущего пользователя (CurrentUser).
4. Системное хранилище **Личное** (**My**) локального компьютера (LocalMachine).

Примечание. После нахождения сертификата в одном из указанных местоположений поиск прекращается.

Если у пользователя нет прав доступа к системному хранилищу LocalMachine, этап 4 не выполняется.

Если при вызове функции `CryptMsgOpenToEncode` и `CryptMsgControl` указан флаг `CMSG_CADES_DISABLE_CERT_SEARCH (=0x00000400)`, то этапы поиска 3 и 4 не выполняются.

После нахождения сертификата выбор атрибута `ESS signing-certificate` или `ESS signing-certificate-v2`, а также его вычисление выполняется в следующем порядке:

1. По открытому ключу, содержащемуся в найденном сертификате, определяется алгоритм электронной подписи.
2. По алгоритму электронной подписи определяется множество алгоритмов хэширования, поддерживаемых этим алгоритмом.
3. Если в найденном множестве присутствует алгоритм хэширования SHA1, то используется атрибут `ESS signing-certificate`, вычисляемый по алгоритму SHA1;
4. Если в найденном множестве нет алгоритма хэширования SHA1, то используется атрибут `ESS signing-certificate-v2`, вычисляемый по первому алгоритму из списка поддерживаемых.

Для поддержки CAdES-BES в функции `CryptEncodeObject(Ex)` используются следующие типы данных:

- `ESS signing-certificate`;
- `ESS signing-certificate-v2`.

Тип структуры ESS signing-certificate

`ITCS_STRUCT_TYPE_ESS_SIGNING_CERTIFICATE_ATTR "{A010E44A-53E3-4e45-ACDB-272FF57EAA8C}"` — константа, используемая в параметре `lpszStructType`. Указывает, что параметр `pvStructInfo` является указателем на структуру типа `ITCS_ESS_SIGNING_CERTCERTIFICATE_ATTR` (см. «Объектные идентификаторы, используемые при формировании расширенной электронной подписи» на стр. 82), соответствующим структуре `id-aa-signingCertificate OBJECT IDENTIFIER ::= { iso(1) member - body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id - aa(2) 12 }`. Подробнее см. RFC 2634 (http://www.ietf.org/rfc/rfc2634) и RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 5.7.3.1.

Тип структуры ESS signing-certificate-v2

`ITCS_STRUCT_TYPE_ESS_SIGNING_CERTCERTIFICATE_ATTR_V2 "{B82662AD-F1FF-42c3-86EC-F9B532E6C4C2}"` — константа, используемая в параметре `lpszStructType`. Указывает, что параметр `pvStructInfo` является указателем на структуру типа `ITCS_ESS_SIGNING_CERTCERTIFICATE_ATTR_V2` (см. «Объектные идентификаторы, используемые при формировании расширенной электронной подписи» на стр. 82), соответствующим структуре `id-aa-signingCertificateV2 OBJECT IDENTIFIER :: = { iso(1) member - body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id - aa(2) 47 }`. Подробнее см. RFC 5035 (http://tools.ietf.org/html/rfc5035), раздел 5.4.1.

Объектные идентификаторы, используемые при формировании расширенной электронной подписи

Таблица 46. Идентификаторы, используемые при формировании расширенной электронной подписи

Идентификатор	Значение	Описание идентификатора
`szOID_RSA_SMIME_SignatureTimeStamp`	`1.2.840.113549.1.9.16.2.14`	Объектный идентификатор, определяющий атрибут `signatureTimeStampToken` (штамп времени) подписанного сообщения. Подробнее см. RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел6.1.1.«
`szOID_ITCS_ESS_SIGNING_CERTIFICATE_ATTR`	`1.2.840.113549.1.9.16.2.12`	Объектный идентификатор, определяющий атрибут `signingCertificate` подписанного сообщения. Подробнее см. RFC 2634 (http://www.ietf.org/rfc/rfc2634) и RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 5.7.3.1.»
`szOID_ITCS_ESS_SIGNING_CERTCERTIFICATE_ATTR_V2`	`1.2.840.113549.1.9.16.2.47`	Объектный идентификатор, определяющий атрибут `signingCertificateV2` подписанного сообщения. Подробнее см. RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 5.7.3.2.«
`szOID_RSA_SMIME_CompleteCertRefs`	`1.2.840.113549.1.9.16.2.21`	Объектный идентификатор, определяющий атрибут `ets-certificateRefs` подписанного сообщения. Подробнее см. RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 6.2.1.
`szOID_RSA_SMIME_CompleteRevocationRefs`	`1.2.840.113549.1.9.16.2.22`	Объектный идентификатор, определяющий атрибут `ets-revocationRefs` подписанного сообщения. Подробнее см. RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 6.2.2.
`szOID_RSA_SMIME_CompleteCertValues`	`1.2.840.113549.1.9.16.2.23`	Объектный идентификатор, определяющий атрибут `ets-certValues` подписанного сообщения. Подробнее см. RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 6.3.3.

Идентификатор	Значение	Описание идентификатора
szOID_RSA_SMIME_CompleteRevocationValues	1.2.840.113549.1.9.16.2.24	Объектный идентификатор, определяющий атрибут ets-revocationValues подписанного сообщения. Подробнее см. RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 6.3.4.
szOID_RSA_SMIME_CadesCTimeStamp	1.2.840.113549.1.9.16.2.25	Объектный идентификатор, определяющий атрибут ets-escTimeStamp подписанного сообщения. Подробнее см. RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 6.3.5.
szOID_RSA_SMIME_CadesCTimeStampedCertsCrlsRefs	1.2.840.113549.1.9.16.2.26	Объектный идентификатор, определяющий атрибут ets-certCRLTimestamp подписанного сообщения. Подробнее см. RFC 5126 (http://tools.ietf.org/html/rfc5126), раздел 6.3.6.

7

Дополнительные параметры функций CryptoAPI

Таблица 47. Дополнительные параметры функций CryptoAPI

Параметр	Описание
CRYPT_PKCS8_KEY_BAG	Флаг для функции CryptExportKey, изменяющий формат экспортируемого блоба на PKCS #8.
CMSG_CADES_STRICT	Дополнительный флаг к функциям CryptMsgOpenToEncode и CryptMsgControl. Если не удалось добавить атрибуты CADES-BES, функции CryptMsgOpenToEncode и CryptMsgControl будут возвращать ошибку. Данный флаг не может быть указан одновременно с флагом CMSG_CADES_DISABLE.
CMSG_CADES_DISABLE	Дополнительный флаг к функциям CryptMsgOpenToEncode и CryptMsgControl. Отключает добавление атрибутов CADES-BES. Данный флаг не может быть указан одновременно с флагом CMSG_CADES_STRICT.
CMSG_CADES_DISABLE_CERT_SEARCH	Дополнительный флаг к функциям CryptMsgOpenToEncode и CryptMsgControl. Отключает поиск сертификата подписывающего в хранилищах «Личное» текущего пользователя и локального компьютера.
CRYPT_MESSAGE_CADES_STRICT	Дополнительный флаг к функции CryptSignMessage. Если не удалось добавить атрибуты CADES-BES, функция CryptSignMessage будет возвращать ошибку. Данный флаг не может быть указан одновременно с флагом CRYPT_MESSAGE_CADES_DISABLE.

Параметр	Описание
CRYPT_MESSAGE_CADES_DISABLE	Дополнительный флаг к функции CryptSignMessage. Отключает добавление атрибутов CADES-BES. Данный флаг не может быть указан одновременно с флагом CRYPT_MESSAGE_CADES_STRICT.

8

Работа с контейнерами ключей на внешних устройствах

Общие сведения

Криптопровайдер ViPNet CSP может идентифицировать имена контейнеров ключей, расположенных на внешних устройствах, если они заданы в одном из следующих форматов:

- Допустимые имена контейнеров ключей (на стр. 88).
- Канонические имена контейнеров ключей (на стр. 89).
- Имена контейнеров ключей в ОС Windows (на стр. 91).

Допустимые имена контейнеров ключей

Чтобы криптопровайдер ViPNet CSP верно идентифицировал контейнер ключей, находящийся на внешнем устройстве, задайте его в одном из следующих форматов:

- `\\!\<Название семейства устройств>\<UID устройства>\<Имя файла контейнера>` — при данном формате задания контейнер ключей будет всегда однозначно идентифицирован.
- `\\!\<Название семейства устройств>\\<Имя файла контейнера>` — при данном формате задания контейнер ключей будет однозначно идентифицирован в случае, если к компьютеру подключено только одно устройство этого семейства.
- `\\!\\\<Имя файла контейнера>` — при данном формате задания контейнер ключей будет однозначно идентифицирован в случае, если к компьютеру подключено только одно устройство.
- `\\!\\\` — данный формат может быть удобен, например, для перечисления контейнеров ключей на всех подключенных к компьютеру устройствах.

Здесь:

- `<Название семейства устройств>` — см. документы «ViPNet CSP. Руководство пользователя», «ViPNet CSP for Linux. Руководство пользователя».
- `<UID устройства>` — уникальный идентификатор устройства. Представляет собой строку из 16 символов (в случае если идентификатор состоит из меньшего числа символов, его необходимо дополнить справа пробелами).

Примечание. Обычно этот идентификатор вы можете увидеть в утилите производителя устройства при подключенном устройстве. Например, если в такой утилите указан идентификатор `0x0054d323`, то `<UID устройства>` необходимо задать следующим образом: «`0054d323 `».

Примеры задания контейнеров ключей:

```
\\!\SafeNet eToken (eToken Aladdin)\0054d278        \container1
\\!\JaCarta\\container2
```

Канонические имена контейнеров ключей

Канонический формат имен контейнеров ключей является внутренним представлением, к которому приводятся все остальные имена контейнеров ключей, заданных во всех остальных форматах.

Примечание. Мы не рекомендуем вам задавать контейнеры ключей, находящиеся на внешних устройствах, в каноническом формате, так как этот формат может различаться в разных версиях ViPNet CSP.

Канонический формат имен контейнеров ключей:

```
1|2\5|<ID семейства устройств>\17|<UID устройства>\8|<Имя файла контейнера>
```

Здесь:

- `<ID семейства устройств>` — идентификатор семейства внешних устройств (см. документы «ViPNet CSP. Руководство пользователя», «ViPNet CSP for Linux. Руководство пользователя»):

  Таблица 48. Идентификаторы семейств внешних устройств в ViPNet CSP

Название семейства устройств в ViPNet CSP	Идентификатор семейства устройств
`SafeNet eToken (eToken Aladdin)`	300
`Rutoken/Rutoken S`	1200
`Rutoken ECP/Rutoken Lite`	1250
`eToken GOST/JaCarta GOST`	1400
`JCDS`	2300
`Infotecs Software Token`	3000
`ViPNet HSM`	2500
`JaCarta`	2350
`ESMART Token`	4000
`A-Key` (поддерживается только в версии ViPNet CSP для ОС Windows)	2700
`UEC` (поддерживается только в версии ViPNet CSP для ОС Windows)	5000

- `<UID устройства>` — уникальный идентификатор устройства (см. »Допустимые имена контейнеров ключей« на стр. 88).

Пример задания контейнера ключей в каноническом формате:

```
1|2\5|300\17|0054d323        \8|container
```

Имена контейнеров ключей в ОС Windows

При взаимодействии криптопровайдера ViPNet CSP с ОС Windows используются имена контейнеров ключей в следующем формате:

```
\\.\<Считыватель>\<Имя контейнера>
```

Здесь:

- `<Считыватель>` — имя считывателя внешних устройств в ОС Windows.
- `<Имя контейнера>` — уникальное имя контейнера ключей в ОС Windows.

9

Кэширование в ViPNet CSP Linux

Назначение кэширования

Для оптимизации операций проверки подлинности сертификатов и построения цепочек сертификации в ViPNet CSP Linux реализовано кэширование результатов некоторых операций. Кэш расположен в оперативной памяти процесса, поэтому каждый из одновременно работающих процессов имеет независимый кэш. Ограничение объема оперативной памяти, выделенной для кэша, настраивается отдельно для каждой операции в конфигурационном файле `csp.ini`, который входит в пакет `itcs-csp-gost`. Общее ограничение рассчитывается как сумма ограничений для каждой операции (см. »Операции, для которых реализовано кэширование« на стр. 94). При превышении допустимого объема памяти для операции из кэша удаляются наиболее давно неиспользованные элементы (в соответствии с алгоритмом LRU).

Операции, для которых реализовано кэширование

В ViPNet CSP Linux кэширование реализовано для следующих операций:

1. Декодирование контекста списка CRL (функция `CertCreateCRLContext`).

   Результат, запоминаемый в кэше, — декодированная структура `CRL_INFO`. Результат идентифицируется в кэш по массиву с закодированным списком CRL. Время нахождения результата в кэше ограничено. Результаты, время хранения которых превышено, удаляются из кэша.

   Параметры, задаваемые в конфигурационном файле ViPNet CSP Linux (`/etc/opt/itcs/vipnet-csp/csp.ini`):

   - `crl_cache_enable` — включение или отключение кэширования для операции. Возможные значения: `yes` — включено, `no` — отключено.
   - `crl_cache_size_mb` — максимальный размер оперативной памяти для кэширования (в мегабайтах).
   - `crl_cache_storage_time_s` — максимальное время хранения объекта в кэше (в секундах).

2. Проверка электронной подписи сертификатов и списков CRL (функция `CryptVerifyCertificateSignature`).

   Результат, запоминаемый в кэше, — факт успешной проверки электронной подписи. Факты неудачных проверок не кэшируются. Результат идентифицируется в кэше по массиву данных, для которых проверялась электронная подпись, массиву электронной подписи и массиву ключа проверки электронной подписи.

   Параметры, задаваемые в конфигурационном файле ViPNet CSP Linux (`/etc/opt/itcs/csp.ini`):

   - `sign_cache_enable` — включение или отключение кэширования для операции. Возможные значения: `yes` — включено, `no` — отключено.
   - `sign_cache_size_mb` — максимальный размер оперативной памяти для кэширования (в мегабайтах) для операции проверки электронной подписи.

Настройки кэширования по умолчанию (определены в файле csp.ini пакета itcs-csp-gost):

```
crl_cache_enable = yes
crl_cache_size_mb = 128
crl_cache_storage_time_s = 600
sign_cache_enable = yes
sign_cache_size_mb = 64
```

A

Ограничения реализации

При добавлении открепленной подписи (см. глоссарий, стр. 96) необходимо использовать алгоритм, применявшийся при формировании первой электронной подписи. Добавление открепленной подписи, сформированной по иному алгоритму, невозможно.

B

Глоссарий

ViPNet Удостоверяющий и ключевой центр (УКЦ)

Программа, входящая в состав программного обеспечения ViPNet Administrator. Администратор УКЦ формирует и обновляет ключи для сетевых узлов ViPNet, а также управляет сертификатами и списками аннулированных сертификатов.

Открепленная подпись

Тип электронной подписи, при использовании которого электронная подпись и служебная информация помещаются в отдельный файл. Далее для проверки электронной подписи требуется не только данный контейнер, но и исходный файл, который в контейнер не входит.

Электронная рулетка

Встроенный компонент программного обеспечения ViPNet, который позволяет инициализировать датчик случайных чисел на основе действий пользователя. Полученная последовательность используется при формировании ключей узла.