ESMART® TOKEN ГОСТ

Руководство программиста ESMART PKCS11

АННОТАЦИЯ

Данный документ описывает особенности реализации и работы с аппаратными модулями СКЗИ «ESMART Token ГОСТ» с использованием прикладного программного интерфейса, соответствующего стандарту PKCS#11.

Средство криптографической защиты информации ESMART Token ГОСТ на базе отечественной микросхемы MIK51SC72DV6.

RU.63793390.00009-01

Первая редакция.

Москва

2017

Содержание

1. Общая информация 4
2. Назначение и условия применения 4
3. Состав и наименование динамических библиотек 4
4. Обращение к библиотеке 4
5. Алгоритм работы 5
5.1 Порядок работы с библиотекой 5
5.2 Порядок работы с токеном/картой 6
6. Объекты 7
6.1 Поддерживаемые типы объектов 7
6.2 Создание объектов 7
6.3 Общие атрибуты объектов 8
6.4 Атрибуты объектов данных CKO_DATA 9
6.5 Атрибуты объектов сертификатов CKO_CERTIFICATE 10
6.6 Общие атрибуты объектов ключей 11
6.7 Общие атрибуты открытых ключей CKO_PUBLIC_KEY 11
6.8 Общие атрибуты закрытых ключей CKO_PRIVATE_KEY 11
6.9 Объект CKO_PUBLIC_KEY, тип CKK_RSA 12
6.10 Объект CKO_PRIVATE_KEY, тип CKK_RSA 13
6.11 Объект CKO_PUBLIC_KEY, тип CKK_ECDSA 14
6.12 Объект CKO_PRIVATE_KEY, тип CKK_ECDSA 15
6.13 Объект CKO_PUBLIC_KEY, тип CKK_GOSTR3410 16
6.14 Объект CKO_PRIVATE_KEY, тип CKK_GOSTR3410 18
6.15 Объект CKO_SECRET_KEY, тип CKK_GOST28147 20
6.16 Атрибуты, которые можно изменить после создания объекта 21
6.17 Значения атрибутов, принимаемые по умолчанию 21
7. Таблица функций 22
7.1 Список функций стандарта PKCS11 22
7.2 Список дополнительных функций 26
8. Собственные функции 28
9. Реализация с особенностями 45
10. Обработка ошибок 46
10.1 Специфические коды, возвращаемые функцией C_ISBC_pkcs7VerifyEx 46

1. Общая информация

Данный документ является руководством разработчика для создания программ с использованием динамической подключаемой библиотеки, предоставляющей высокоуровневый прикладной программный интерфейс. Текущая реализация интерфейса соответствует стандарту OASIS Open PKCS #11 Cryptographic Token Interface Base Specification Version 2.40 (Criptoki)» (далее PKCS#11). Поддержка криптографических алгоритмов ГОСТ 28147-89, ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012, ГОСТ Р 34.11-94, ГОСТ Р 34.11-2012 выполнена с учетом методических рекомендаций Технического комитета ТК 26 «Расширение стандарта PKCS#11 для использования российских криптографических алгоритмов».

2. Назначение и условия применения

Библиотеки на языке C++ предоставляют высокоуровневый прикладной программный интерфейс (API) соответствующий стандарту PKCS#11 для вызова криптографических и вспомогательных функций аппаратных модулей СКЗИ ESMART Token ГОСТ.

В состав СКЗИ ESMART Token ГОСТ входят программные библиотеки для операционных систем GNU/Linux. Полный список поддерживаемых версий операционных систем приведен в формуляре на СКЗИ. В зависимости от разрядности процессора и операционной системы предусмотрено наличие 32-разрядных и 64-разрядных библиотек.

Библиотеки предоставляются в скомпилированном виде.

3. Состав и наименование динамических библиотек

Название	*ОС*	*Описание*
libisbc_pkcs11_main.so	*GNU/Linux*	*Основная библиотека, реализующая основные функции взаимодействия с картой/токеном по стандарту PKCS#11 (Cryptoki).*
libesmart_token_gost_mod.so	*GNU/Linux*	*Библиотека-модуль для работы с картами/токенами ESMART Token ГОСТ.*

Способы установки библиотек описаны в Руководстве администратора ESMART PKCS11 RU.63793390.00009-01 92.

4. Обращение к библиотеке

Для работы с библиотекой из приложений C# рекомендуется использовать библиотеку-оболочку с открытым исходным кодом Net.Sf.Pkcs11.

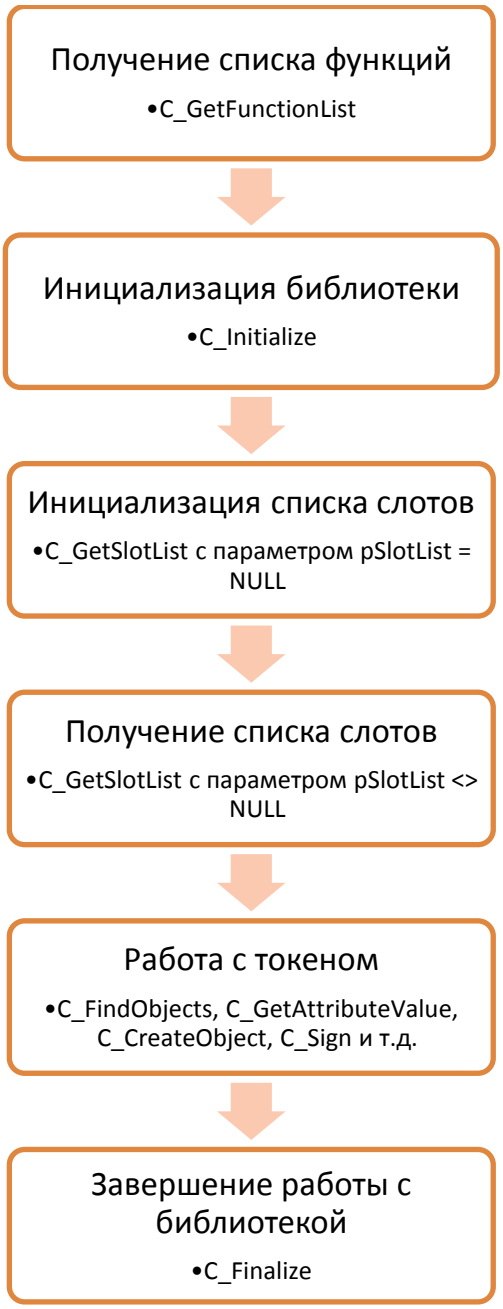
Для работы с библиотекой из приложений на языке Java используется Sun PKCS11 Provider. Sun PKCS11 Provider не поддерживает работу с отечественными алгоритмами. Проприетарные функции, приведенные в руководстве, и механизмы ГОСТ по ТК26 для стандартных функций PKCS11 через Sun PKCS11 Provider не поддерживаются.

Для вызова функций библиотеки из приложений на C++ библиотек-оболочек не требуется.

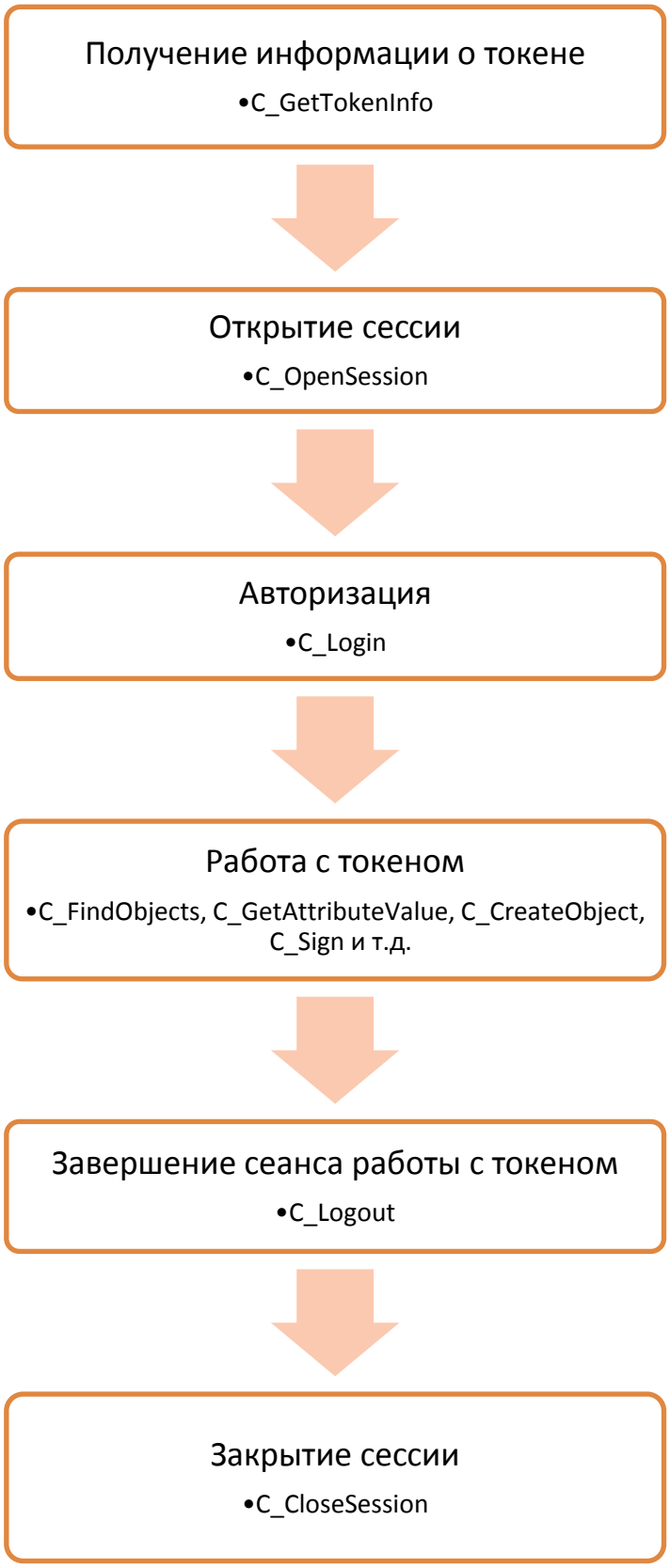
5. Алгоритм работы

5.1 Порядок работы с библиотекой

Работа с библиотекой должна начинаться с вызова функции C_Initialize для инициализации библиотеки и должна завершаться вызовом C_Finalize.



5.2 Порядок работы с токеном/картой



6. Объекты

6.1 Поддерживаемые типы объектов

Библиотека поддерживает объекты следующих типов:

`CKO_DATA` *(данные),*
`CKO_CERTIFICATE` *(сертификат формата X.509),*
`CKO_PUBLIC_KEY` *(открытые ключи RSA, ГОСТ Р 34.10 и ECDSA) ,*
`CKO_PRIVATE_KEY` *(закрытые ключи RSA, ГОСТ Р 34.10 и ECDSA) ,*
`CKO_SECRET_KEY` *(ключи симметричного шифрования DES, 3DES, AES и ГОСТ 28147-89) .*

Объекты `CKO_PUBLIC_KEY` *защищены PIN-кодом пользователя только на запись при создании. Соответственно, чтение значения возможно без предъявления PIN-кода, в том числе, для объектов, созданных с атрибутом* `CKA_PRIVATE`.

Объекты `CKO_CERTIFICATE` *с атрибутом* `CKA_TRUSTED` *могут быть загружены только по предъявлению SO PIN.*

6.2 Создание объектов

Для создания объектов используются функции ***C_CreateObject***, ***C_GenerateKey***, ***C_GenerateKeyPair***, ***C_UnwrapKey*** *и* ***C_DeriveKey***. *Создание объектов с помощью стандартной функции* ***C_CopyObject*** *не поддерживается.*

Для создания объекта необходимо предоставить функции соответствующий шаблон объекта.

Возможные ошибки при передаче атрибутов:

`CKR_ATTRIBUTE_TYPE_INVALID`	*передан неподдерживаемый атрибут*
`CKR_ATTRIBUTE_VALUE_INVALID`	*передано некорректное значение атрибута*
`CKR_ATTRIBUTE_READ_ONLY`	*если предоставленный шаблон определяет значение для атрибута «только для чтения», то объект не создается*
`CKR_TEMPLATE_INCOMPLETE`	*если значений атрибута в предоставленном шаблоне вместе со значениями по умолчанию и значениями, задаваемыми функцией при создании объекта, оказывается недостаточно для полного определения объекта, то объект не создается*
`CKR_TEMPLATE_INCONSISTENT`	*если значения атрибута в поддерживаемом шаблоне вместе со значениями по умолчанию и значениями, задаваемыми функцией при создании объекта, являются несовместимыми, то объект не создается*

6.3 Общие атрибуты объектов

Атрибут	Тип данных	Значение
CKA_CLASS	CK_OBJECT_CLASS	должен быть определен при создании объекта с помощью функции **C_CreateObject.**
CKA_TOKEN	CK_BBOOL	CK_TRUE, если объект является объектом токена (т.е. будет создан на токене) CK_FALSE, если объект является объектом сессии (т.е. будет создан в оперативной памяти) [1]
CKA_PRIVATE	CK_BBOOL	CK_TRUE, если это закрытый объект. Доступ к объекту можно получить только после ввода PIN-кода пользователя CK_FALSE, если это открытый объект. Доступ к объекту можно получить без ввода PIN-кода пользователя
CKA_MODIFIABLE	CK_BBOOL	CK_TRUE, если значения атрибутов объекта после его создания могут быть изменены. По умолчанию принимает значение CK_TRUE. CK_FALSE, если объект создан «только для чтения» и значения атрибутов объекта после его создания не могут быть изменены[2]
CKA_LABEL	RFC2279 string	Метка объекта (по умолчанию метка пуста)[3]

1. Сеансовые объекты (`CKA_TOKEN=CK_FALSE`) поддерживаются только для `CKO_SECRET_KEY` и `CKO_PUBLIC_KEY`.
2. Запрет изменений установкой `CKA_MODIFIABLE=CK_FALSE` действует только на атрибут `CKA_VALUE` для объектов данных `CKO_DATA`. При работе с объектами других типов и их атрибутами значение `CKA_MODIFIABLE` игнорируется.
3. `CKA_LABEL` может быть изменен после создания объекта.

6.4 Атрибуты объектов данных CKO_DATA

Атрибут	Тип данных	Значение
CKA_APPLICATION	RFC2279 string	Приложение, для которого создан объект данных. По умолчанию значение пустое
CKA_OBJECT_ID	Byte Array	Идентификатор объекта в DER-кодировке, обозначающий тип объекта данных (по умолчанию пусто)
CKA_VALUE	Byte Array	Значение объекта (по умолчанию пусто)[1]

1. Значение `CKA_VALUE` можно сделать неизменяемым после создания, если применить модификатор `CKA_MODIFIABLE=CK_FALSE`. Атрибут `CKA_MODIFIABLE` задается только при создании объекта и не может быть изменен в дальнейшем.

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_DATA;
CK_UTF8CHAR label[] = "A data object";
CK_UTF8CHAR application[] = "An application";
CK_BYTE data[] = "Sample data";
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_TOKEN, &true_, sizeof(true_)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_APPLICATION, application, sizeof(application)-1},
      {CKA_VALUE, data, sizeof(data)}
};
```

6.5 Атрибуты объектов сертификатов CKO_CERTIFICATE

Атрибут	Тип данных	Значение
CKA_CERTIFICATE_TYPE	CK_CERTIFICATE_TYPE	Тип сертификата
CKA_TRUSTED	CK_BBOOL	Сертификат является доверенным для приложения, которым был создан. Создание сертификата с атрибутом `CKA_TRUSTED` требует SO PIN
CKA_CERTIFICATE_CATEGORY	CK_ULONG	Значение атрибута будет возвращаться при чтении, только если атрибут был задан при создании объекта. Если при создании объекта атрибут не был задан, при попытке его чтения возвращается ошибка `CKR_ATTRIBUTE_INVALID`
CKA_CHECK_VALUE	Byte array	Контрольная сумма
CKA_START_DATE	CK_DATE	Дата начала действия (по умолчанию пусто)
CKA_END_DATE	CK_DATE	Дата окончания действия (по умолчанию пусто)
CKA_SUBJECT	Byte Array	Имя Subject сертификата в кодировке DER[1]
CKA_ID	Byte array	Идентификатор ключевой пары (по умолчанию пусто)
CKA_ISSUER	Byte array	Имя издателя сертификата в кодировке DER[1]
CKA_SERIAL_NUMBER	Byte array	Серийный номер сертификата в кодировке DER[1]
CKA_VALUE	Byte array	Значение в кодировке **BER**

1. *Переданные значения атрибутов CKA_SUBJECT, CKA_ISSUER, CKA_SERIAL_NUMBER при создании игнорируются и подставляются автоматически из переданного сертификата.*

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_CERTIFICATE;
CK_CERTIFICATE_TYPE certType = CKC_X_509;
CK_UTF8CHAR label[] = "A certificate object";
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BYTE certificate[] = {...};
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_CERTIFICATE_TYPE, &certType, sizeof(certType)};
      {CKA_TOKEN, &true_, sizeof(true_)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_SUBJECT, subject, sizeof(subject)},
      {CKA_ID, id, sizeof(id)},
      {CKA_VALUE, certificate, sizeof(certificate)}
};
```

6.6 Общие атрибуты объектов ключей

Атрибут	Тип данных	Значение
CKA_KEY_TYPE	CK_KEY_TYPE	Тип ключа
CKA_ID	Byte array	Идентификатор ключа. По умолчанию пусто
CKA_START_DATE	CK_DATE	Дата начала действия ключа (по умолчанию пусто)
CKA_END_DATE	CK_DATE	Дата окончания действия ключа (по умолчанию пусто)
CKA_DERIVE	CK_BBOOL	CK_TRUE, если ключ поддерживает деривацию (на базе исходного ключа можно создать производные ключи). По умолчанию принимает значение CK_FALSE
CKA_LOCAL	CK_BBOOL	CK_TRUE, если ключ был сгенерирован на токене вызовом функций **C_GenerateKey** или **C_GenerateKeyPair**

6.7 Общие атрибуты открытых ключей CKO_PUBLIC_KEY

Атрибут	Тип данных	Значение
CKA_ENCRYPT	CK_BBOOL	CK_TRUE, если ключ поддерживает шифрование
CKA_VERIFY	CK_BBOOL	CK_TRUE, если ключ поддерживает проверку подписи, представленной в виде приложения к данным
CKA_VERIFY_RECOVER	CK_BBOOL	CK_TRUE, проверка подписи с восстановлением данных[1]
CKA_WRAP	CK_BBOOL	CK_TRUE, если ключ поддерживает шифрование других ключей
CKA_TRUSTED	CK_BBOOL	Ключ может быть доверенным для приложения, которым был создан

1. Поддерживается только для алгоритма RSA

6.8 Общие атрибуты закрытых ключей CKO_PRIVATE_KEY

Атрибут	Тип данных	Значение
CKA_SENSITIVE	CK_BBOOL	CK_TRUE, если ключ является чувствительным (не может быть извлечен из токена в открытом виде)[1]
CKA_DECRYPT	CK_BBOOL	CK_TRUE, если ключ поддерживает расшифрование
CKA_SIGN	CK_BBOOL	CK_TRUE, если ключ поддерживает формирование подписи, которая представлена в виде приложения к данным
CKA_SIGN_RECOVER	CK_BBOOL	CK_TRUE, если ключ поддерживает формирование подписи с восстановлением данных
CKA_UNWRAP	CK_BBOOL	CK_TRUE, если ключ поддерживает расшифрование ключей
CKA_EXTRACTABLE	CK_BBOOL	CK_TRUE, если ключ является извлекаемым и может быть зашифрован[1]
CKA_ALWAYS_SENSITIVE	CK_BBOOL	CK_TRUE, если ключ всегда имеет значение атрибута CKA_SENSITIVE равным CK_TRUE[1]
CKA_NEVER_EXTRACTABLE	CK_BBOOL	CK_TRUE, если ключ никогда не имеет значение атрибута CKA_EXTRACTABLE равным CK_TRUE[1]

1. *Объекты закрытых и симметричных ключей никогда не могут быть извлечены (прочтены или скопированы) с токена/карты независимо от значения параметра CKA_SENSITIVE и CKA_EXTRACTABLE. Извлекаемыми могут быть только ключи, выработанные в процессе VKO.*

6.9 Объект CKO_PUBLIC_KEY, тип CKK_RSA

Атрибут	Тип данных	Значение
CKA_MODULUS	Big Integer	Модуль n
CKA_MODULUS_BITS	CK_ULONG	Длина модуля n в битах
CKA_PUBLIC_EXPONENT	Big integer	Открытая экспонента e

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_UTF8CHAR label[] = "An RSA public key object";
CK_BYTE modulus[] = {...};
CK_BYTE exponent[] = {...};
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
      {CKA_TOKEN, &true, sizeof(true)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_WRAP, &true_, sizeof(true_)},
      {CKA_ENCRYPT, &true, sizeof(true)},
      {CKA_MODULUS, modulus, sizeof(modulus)},
      {CKA_PUBLIC_EXPONENT, exponent, sizeof(exponent)}
};
```

6.10 Объект CKO_PRIVATE_KEY, тип CKK_RSA

Атрибут	Тип данных	Значение
CKA_MODULUS	Big integer	Модуль n
CKA_PUBLIC_EXPONENT	Big integer	Открытая экспонента e
CKA_PRIVATE_EXPONENT	Big integer	Закрытая экспонента d
CKA_PRIME_1	Big integer	Простое число p
CKA_PRIME_2	Big integer	Простое число q
CKA_EXPONENT_1	Big integer	Закрытая экспонента d по модулю p-1
CKA_EXPONENT_2	Big integer	Закрытая экспонента d по модулю q-1
CKA_COEFFICIENT	Big integer	Коэффициент CTR RSA q^{-1} по модулю p

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_UTF8CHAR label[] = "An RSA private key object";
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BYTE modulus[] = {...};
CK_BYTE publicExponent[] = {...};
CK_BYTE privateExponent[] = {...};
CK_BYTE prime1[] = {...};
CK_BYTE prime2[] = {...};
CK_BYTE exponent1[] = {...};
CK_BYTE exponent2[] = {...};
CK_BYTE coefficient[] = {...};
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
      {CKA_TOKEN, &true_, sizeof(true_)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_SUBJECT, subject, sizeof(subject)},
      {CKA_ID, id, sizeof(id)},
      {CKA_SENSITIVE, &true_, sizeof(true_)},
      {CKA_DECRYPT, &true_, sizeof(true_)},
      {CKA_SIGN, &true_, sizeof(true_)},
      {CKA_MODULUS, modulus, sizeof(modulus)},
      {CKA_PUBLIC_EXPONENT, publicExponent, sizeof(publicExponent)},
      {CKA_PRIVATE_EXPONENT, privateExponent, sizeof(privateExponent)},
      {CKA_PRIME_1, prime1, sizeof(prime1)},
      {CKA_PRIME_2, prime2, sizeof(prime2)},
      {CKA_EXPONENT_1, exponent1, sizeof(exponent1)},
      {CKA_EXPONENT_2, exponent2, sizeof(exponent2)},
      {CKA_COEFFICIENT, coefficient, sizeof(coefficient)}
};
```

6.11 Объект CKO_PUBLIC_KEY, тип CKK_ECDSA

Атрибут	Тип данных	Значение
CKA_EC_PARAMS	Byte array	Значение ANSI X9.62 Parameters Value в кодировке DER[1]
CKA_EC_POINT	Byte array	Значение ANSI X9.62 ECPoint value Q в кодировке DER

1. *Значение параметра CKA_EC_PARAMS*

```
const BYTE id_ecdsa_prime256v1
      [] = { 0x06, 0x08, 0x2A, 0x86, 0x48, 0xCE, 0x3D, 0x03, 0x01, 0x07 }
```

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_EC;
CK_UTF8CHAR label[] = "An EC public key object";
CK_BYTE ecParams[] = {...};
CK_BYTE ecPoint[] = {...};
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
      {CKA_TOKEN, &true_, sizeof(true_)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_EC_PARAMS, ecParams, sizeof(ecParams)},
      {CKA_EC_POINT, ecPoint, sizeof(ecPoint)}
};
```

6.12 Объект CKO_PRIVATE_KEY, тип CKK_ECDSA

Атрибут	Тип данных	Значение
CKA_EC_PARAMS	Byte array	Значение ANSI X9.62 Parameters Value в кодировке DER
CKA_VALUE	Big integer	Значение ANSI X9.62 private value d

1. *Значение параметра CKA_EC_PARAMS*

```
const BYTE id_ecdsa_prime256v1
      [] = { 0x06, 0x08, 0x2A, 0x86, 0x48, 0xCE, 0x3D, 0x03, 0x01, 0x07 }
```

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_EC;
CK_UTF8CHAR label[] = "An EC private key object";
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BYTE ecParams[] = {...};
CK_BYTE value[] = {...};
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
      {CKA_TOKEN, &true_, sizeof(true_)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_SUBJECT, subject, sizeof(subject)},
      {CKA_ID, id, sizeof(id)},
      {CKA_SENSITIVE, &true_, sizeof(true_)},
      {CKA_DERIVE, &true_, sizeof(true_)},
      {CKA_EC_PARAMS, ecParams, sizeof(ecParams)},
      {CKA_VALUE, value, sizeof(value)}
};
```

6.13 Объект CKO_PUBLIC_KEY, тип CKK_GOSTR3410

Атрибут	Тип данных	Значение
CKA_VALUE	Byte array	64 байта открытого ключа; по 32 байта на каждую координату X и Y точки эллиптических кривых P(X, Y) в порядке, начиная с младшего байта (little endian)
CKA_GOSTR3410_PARAMS	Byte array	Идентификатор, указывающий на тип объекта данных ГОСТ Р 34.10-2001 (OID парамсета) в DER-кодировке[1]
CKA_GOSTR3411_PARAMS	Byte array	Идентификатор, указывающий на тип объекта данных ГОСТ Р 34.11-94 (OID парамсета) в DER-кодировке[2]
CKA_GOST28147_PARAMS	Byte array	Идентификатор, указывающий на тип объекта данных ГОСТ 28147-89 (OID парамсета) в DER-кодировке. Значение атрибута может быть пропущено

1. Идентификатор CKA_GOSTR3410_PARAMS

```
const BYTE id_GostR3410_2001_TestParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x00 };
const BYTE id_GostR3410_2001_CryptoPro_A_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x01 };
const BYTE id_GostR3410_2001_CryptoPro_B_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x02 };
const BYTE id_GostR3410_2001_CryptoPro_C_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x03 };
const BYTE id_GostR3410_2001_CryptoPro_XchA_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x24, 0x00 };
const BYTE id_GostR3410_2001_CryptoPro_XchB_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x24, 0x01 };
```

2. Идентификатор CKA_GOSTR3411_PARAMS

```
const BYTE id_GostR3411_94_CryptoProParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1E, 0x01 };
const BYTE id_GostR3411_12_256_ParamSet
      [] = { 0x06, 0x08, 0x2a, 0x85, 0x03, 0x07, 0x01, 0x01, 0x02, 0x02 };
```

3. Идентификатор CKA_GOST28147_PARAMS

```
const BYTE id_Gost28147_89_CryptoPro_A_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x01 };
const BYTE id_Gost28147_89_CryptoPro_B_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x02 };
const BYTE id_Gost28147_89_CryptoPro_C_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x03 };
const BYTE id_Gost28147_89_CryptoPro_D_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x04 };
const BYTE id_Gost28147_89_CryptoPro_RIC_1_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x07 };
```

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_GOSTR3410;
CK_UTF8CHAR label[] = “A GOST R34.10 public key object”;
CK_BYTE gostR3410params_oid[] =
{0x06, 0x07, 0x2a, 0x85, 0x03, 0x02, 0x02, 0x23, 0x00};
CK_BYTE gostR3411params_oid[] =
{0x06, 0x07, 0x2a, 0x85, 0x03, 0x02, 0x02, 0x1e, 0x00};
CK_BYTE gost28147params_oid[] =
{0x06, 0x07, 0x2a, 0x85, 0x03, 0x02, 0x02, 0x1f, 0x00};
CK_BYTE value[64] = {...};
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
      {CKA_TOKEN, &true_, sizeof(true_)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_GOSTR3410_PARAMS, gostR3410params_oid,
      sizeof(gostR3410params_oid)},
      {CKA_GOSTR3411_PARAMS, gostR3411params_oid,
      sizeof(gostR3411params_oid)},
      {CKA_GOST28147_PARAMS, gost28147params_oid,
      sizeof(gost28147params_oid)},
      {CKA_VALUE, value, sizeof(value)}
};
```

6.14 Объект CKO_PRIVATE_KEY, тип CKK_GOSTR3410

Атрибут	Тип данных	Значение
CKA_VALUE	Byte array	Закрытый ключ длиной 32 байта в порядке, начиная с младшего байта (little endian)
CKA_GOSTR3410_PARAMS	Byte array	Идентификатор, указывающий на тип объекта данных ГОСТ Р 34.10-2001 (OID парамсета) в DER-кодировке[1]
CKA_GOSTR3411_PARAMS	Byte array	Идентификатор, указывающий на тип объекта данных ГОСТ Р 34.11-94 (OID парамсета) в DER-кодировке[2]
CKA_GOST28147_PARAMS	Byte array	Идентификатор, указывающий на тип объекта данных ГОСТ 28147-89 (OID парамсета) в DER-кодировке[3]

1. *Идентификатор CKA_GOSTR3410_PARAMS*

```
const BYTE id_GostR3410_2001_TestParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x00 };
const BYTE id_GostR3410_2001_CryptoPro_A_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x01 };
const BYTE id_GostR3410_2001_CryptoPro_B_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x02 };
const BYTE id_GostR3410_2001_CryptoPro_C_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x03 };
const BYTE id_GostR3410_2001_CryptoPro_XchA_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x24, 0x00 };
const BYTE id_GostR3410_2001_CryptoPro_XchB_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x24, 0x01 };
```

2. *Идентификатор CKA_GOSTR3411_PARAMS*

```
const BYTE id_GostR3411_94_CryptoProParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1E, 0x01 };
const BYTE id_GostR3411_12_256_ParamSet
      [] = { 0x06, 0x08, 0x2a, 0x85, 0x03, 0x07, 0x01, 0x01, 0x02, 0x02 };
```

3. *Идентификатор CKA_GOST28147_PARAMS*

```
const BYTE id_Gost28147_89_CryptoPro_A_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x01 };
const BYTE id_Gost28147_89_CryptoPro_B_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x02 };
const BYTE id_Gost28147_89_CryptoPro_C_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x03 };
const BYTE id_Gost28147_89_CryptoPro_D_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x04 };
const BYTE id_Gost28147_89_CryptoPro_RIC_1_ParamSet
      [] = { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x07 };
```

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_GOSTR3410;
CK_UTF8CHAR label[] = “A GOST R34.10 private key object”;
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BYTE gostR3410params_oid[] = {0x06, 0x07, 0x2a, 0x85, 0x03, 0x02,
0x02, 0x23, 0x00};
CK_BYTE gostR3411params_oid[] = {0x06, 0x07, 0x2a, 0x85, 0x03, 0x02,
0x02, 0x1e, 0x00};
CK_BYTE gost28147params_oid[] = {0x06, 0x07, 0x2a, 0x85, 0x03, 0x02,
0x02, 0x1f, 0x00};
CK_BYTE value[32] = {...};
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
      {CKA_TOKEN, &true_, sizeof(true_)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_SUBJECT, subject, sizeof(subject)},
      {CKA_ID, id, sizeof(id)},
      {CKA_SENSITIVE, &true_, sizeof(true_)},
      {CKA_SIGN, &true_, sizeof(true_)},
      {CKA_GOSTR3410_PARAMS, gostR3410params_oid,
      sizeof(gostR3410params_oid)},
      {CKA_GOSTR3411_PARAMS, gostR3411params_oid,
      sizeof(gostR3411params_oid)},
      {CKA_GOST28147_PARAMS, gost28147params_oid,
      sizeof(gost28147params_oid)},
      {CKA_VALUE, value, sizeof(value)}
};
```

6.15 Объект CKO_SECRET_KEY, тип CKK_GOST28147

Атрибут	Тип данных	Значение
CKA_VALUE	Byte array	Тело ключа размером в 32 байта, записанное в порядке, начиная со младшего байта
CKA_GOST28147_PARAMS	Byte array	Идентификатор типа объекта данных ГОСТ 28147 (OID параметра) в DER-кодировке[1]

1. Идентификатор CKA_GOST28147_PARAMS

```
const BYTE id_Gost28147_89_CryptoPro_A_ParamSet
    [] =    { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x01 };
const BYTE id_Gost28147_89_CryptoPro_B_ParamSet
    [] =    { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x02 };
const BYTE id_Gost28147_89_CryptoPro_C_ParamSet
    [] =    { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x03 };
const BYTE id_Gost28147_89_CryptoPro_D_ParamSet
    [] =    { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x04 };
const BYTE id_Gost28147_89_CryptoPro_RIC_1_ParamSet
     [] =    { 0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x1F, 0x07 };
```

Пример шаблона, содержащего атрибуты для данного типа объекта:

```
CK_OBJECT_CLASS class = CKO_SECRET_KEY;
CK_KEY_TYPE keyType = CKK_GOST28147;
CK_UTF8CHAR label[] = "A GOST 28147-89 secret key object";
CK_BYTE value[32] = {...};
CK_BYTE params_oid[] = {0x06, 0x07, 0x2a, 0x85, 0x03, 0x02, 0x02,
0x1f, 0x00};
CK_BBOOL true_ = CK_TRUE;
CK_ATTRIBUTE template[] = {
      {CKA_CLASS, &class, sizeof(class)},
      {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
      {CKA_TOKEN, &true_, sizeof(true_)},
      {CKA_LABEL, label, sizeof(label)-1},
      {CKA_ENCRYPT, &true_, sizeof(true_)},
      {CKA_GOST28147_PARAMS, params_oid,sizeof(params_oid)},
      {CKA_VALUE, value, sizeof(value)}
};
```

6.16 Атрибуты, которые можно изменить после создания объекта

Тип объекта / Атрибут	CKO_DATA	CKO_CERTIFICATE	CKO_PUBLIC_KEY	CKO_PRIVATE_KEY	CKO_SECRET_KEY
CKA_LABEL	Изменяемый	Изменяемый	Изменяемый	Изменяемый	Изменяемый
CKA_ID	Не имеет	Изменяемый	Изменяемый	Изменяемый	Изменяемый
CKA_VALUE	Изменяемый[1]	Не изменяемый	Не изменяемый	Не изменяемый	Не изменяемый

6.17 Значения атрибутов, принимаемые по умолчанию

CKA_PRIVATE	CK_FALSE
CKA_TOKEN	CK_FALSE
CKA_MODIFIABLE	CK_TRUE
CKA_ENCRYPT	CK_FALSE
CKA_DECRYPT	CK_FALSE
CKA_SIGN	CK_FALSE
CKA_VERIFY	CK_FALSE
CKA_WRAP	CK_FALSE
CKA_EXTRACTABLE	CK_FALSE
CKA_UNWRAP	CK_FALSE
CKA_DERIVE	CK_FALSE
CKA_TRUSTED	CK_FALSE
CKA_LOCAL	CK_TRUE
CKA_ALWAYS_SENSITIVE	CK_TRUE
CKA_NEVER_EXTRACTABLE	CK_TRUE
CKA_SENSITIVE	CK_TRUE
CKA_ALWAYS_AUTHENTICATE	CK_FALSE
CKA_SIGN_RECOVER	CK_FALSE
CKA_VERIFY_RECOVER	CK_FALSE

7. Таблица функций

Модулем экспортируются функции, соответствующие стандарту PKCS#11 v2.40. Проприетарные функции представлены в отдельном разделе.

7.1 Список функций стандарта PKCS11

GENERAL-PURPOSE FUNCTIONS/Функции общего назначения		
C_Initialize	Инициализация библиотеки	
C_Finalize	Вызывается при завершении работы с библиотекой	
C_GetInfo	Получение информации о библиотеке, например, версия и наименование разработчика	
C_GetFunctionList	Получение адресов функций	
SLOT AND TOKEN MANAGEMENT FUNCTIONS/Функции работы со слотами и токенами		
C_GetSlotList	Получение списка слотов	
C_GetSlotInfo	Получение информации о слоте	
C_GetTokenInfo	Получение информации о токене	
C_WaitForSlotEvent	Ожидание события «прихода/ухода» токена	
C_GetMechanismList	Получение списка поддерживаемых механизмов	
C_GetMechanismInfo	Получение параметров механизма	
C_InitToken	Инициализация токена	В процессе инициализации устанавливается PIN-код пользователя, что не соответствует спецификации и сделано для совместимости с некоторыми существующими решениями.
C_InitPIN	Инициализация PIN-кода пользователя	Функция предназначена для смены или разблокировки (путем смены) ПИН-кода пользователя при предъявлении SO PIN. Может использоваться и в случае, если карта заблокирована и не заблокирована.
C_SetPIN	Установка нового PIN-кода администратора или пользователя	Смена ПИН-кода пользователя, при условии, что пользователь знает свой ПИН-код и карта не заблокирована. Не требует предъявления SO PIN.
C_OpenSession	Открытие сессии	
C_CloseSession	Закрытие сессии	
C_CloseAllSessions	Закрытие всех сессий	

C_GetSessionInfo	*Получение информации о сессии*	
C_GetOperationState[1]	*Получение текущего состояния крипто-операций*	
C_SetOperationState	*Установка состояния крипто-операций*	
C_Login	*Аутентификация*	
C_Logout	*Деаутентификация*	
OBJECT MANAGEMENT FUNCTIONS/Функции управления объектами		
C_CreateObject	*Создание объектов*	*После создания объекта CKO_PRIVATE_KEY следует создавать объект CKO_PUBLIC_KEY*
C_CopyObject[2]	*Копирование объектов*	
C_DestroyObject	*Удаление объектов*	
C_GetObjectSize	*Получение примерного размера памяти, занимаемой объектом*	
C_GetAttributeValue	*Чтение значения атрибута*	
C_SetAttributeValue	*Установка значения атрибута*	
C_FindObjectsInit	*Функции поиска объектов*	
C_FindObjects		
C_FindObjectsFinal		
ENCRYPTION FUNCTIONS/Функции шифрования		
C_EncryptInit	*Функции шифрования*	*Поддерживаемые механизмы:* CKM_RSA_PKCS CKM_RSA_X_509 CKM_AES_ECB CKM_AES_CBC CKM_AES_CBC_PAD CKM_DES_ECB CKM_DES_CBC CKM_DES_CBC_PAD CKM_DES3_ECB CKM_DES3_CBC CKM_DES3_CBC_PAD CKM_GOST28147[3] CKM_GOST28147_ECB
C_Encrypt		
C_EncryptUpdate[4]		
C_EncryptFinal[4]		
DECRYPTION FUNCTIONS/Функции расшифрования		
C_DecryptInit	*Функции дешифрования*	*Поддерживаемые механизмы, как для функции шифрования*
C_Decrypt		
C_DecryptUpdate[4]		
C_DecryptFinal[4]		

[1] *Функции не поддерживаются*

[2] *Функции не поддерживаются*

[3] *В режиме CFB передается механизм CKM_GOST28147*

[4] *В режиме CFB не поддерживаются функции C_EncryptUpdate, C_EncryptFinal, C_DecryptUpdate, C_DecryptFinal*

MESSAGE DIGESTING FUNCTIONS/Функции вычисления хэш-значений		
C_DigestInit C_Digest C_DigestUpdate C_DigestFinal	*Функции вычисления хэш-значений*	*Поддерживаемые механизмы:* CKM_MD5 CKM_SHA_1 CKM_SHA256 CKM_GOSTR3411 CKM_GOSTR3411_12_256
C_DigestKey	*Вычисление хэш-значения секретного ключа*	
SIGNING AND MACING FUNCTIONS/Функции подписи и имитозащиты		
C_SignInit C_Sign C_SignUpdate C_SignFinal C_SignRecoverInit[8] C_SignRecover	*Функции вычисления цифровой подписи*	*Поддерживаемые механизмы:* CKM_RSA_PKCS CKM_GOSTR3410[5] CKM_GOSTR3410_WITH_GOSTR3411[6] CKM_GOSTR3410_WITH_GOSTR3411_12_256[7] CKM_ECDSA CKM_GOST28147_MAC
FUNCTIONS FOR VERIFYING SIGNATURES AND MACS/ Функции проверки подписи и имитозащиты		
C_VerifyInit C_Verify C_VerifyUpdate C_VerifyFinal C_VerifyRecoverInit[9] C_VerifyRecover	*Функции проверки цифровой подписи*	*Поддерживаемые механизмы:* CKM_RSA_PKCS CKM_GOSTR3410 CKM_GOSTR3410_WITH_GOSTR3411 CKM_GOSTR3410_WITH_GOSTR3411_12_256 CKM_ECDSA CKM_GOST28147_MAC
DUAL-FUNCTION CRYPTOGRAPHIC FUNCTIONS/Функции двойного назначения		
C_DigestEncryptUpdate C_DecryptDigestUpdate C_SignEncryptUpdate C_DecryptVerifyUpdate	*Функции, объединяющие выполнение других крипто-функций*	

[5] *Механизм для генерации и проверки ЭП использует ранее вычисленное значение хэш-функции. Ограничение по размеру передаваемых данных составляет 32 байта*

[6] *Механизм для генерации и проверки ЭП (ГОСТ Р 34.10-2001) с хэшированием (ГОСТ Р 34.11-94). Размер подаваемых данных не ограничен*

[7] *Механизм для генерации и проверки ЭП (ГОСТ Р 34.10-2012 в 256-битном режиме) с хэшированием (ГОСТ Р 34.11-2012 в 256-битном режиме). Размер подаваемых данных не ограничен*

[8] *C_SignRecover и C_SignRecoverInit поддерживаются только для алгоритма RSA*

[9] *C_VerifyRecoverInit и C_VerifyRecover поддерживаются только для алгоритма RSA*

KEY MANAGEMENT FUNCTIONS/Функции управления ключами		
C_GenerateKey	*Генерация секретного ключа*	*Поддерживаемые механизмы:* CKM_AES_KEY_GEN CKM_DES_KEY_GEN CKM_DES2_KEY_GEN CKM_GOST28147_KEY_GEN
C_GenerateKeyPair	*Генерация ключевой пары*	*Поддерживаемые механизмы:* CKM_RSA_PKCS_KEY_PAIR_GEN CKM_EC_KEY_PAIR_GEN CKM_GOSTR3410_KEY_PAIR_GEN CKM_GOSTR3410_256_KEY_PAIR_GEN
C_WrapKey	*Шифрование секретного или закрытого ключа* [10]	
C_UnwrapKey	*Расшифрование и создание нового секретного или закрытого ключа*[11]	*Поддерживаемые механизмы:* CKM_GOST28147_KEY_WRAP *в функцию* C_UnwrapKey *передается зашифрованная на* hUnwrappingKey *(*CKK_GOST28147*) структура PKCS#8 для PrivateKeyInfo ASN.1, представленная согласно дополнения ТК26* *После создания объекта* CKO_PRIVATE_KEY *следует создавать объект* CKO_PUBLIC_KEY
C_DeriveKey	*Создание нового ключа на основе существующего*	*Поддерживаемые механизмы:* CKM_GOSTR3410_DERIVE CKM_GOSTR3410_12_DERIVE
RANDOM NUMBER GENERATION FUNCTIONS/Функции выработки случайного числа		
C_SeedRandom	*Установка seed-значения для генератора случайных чисел*	*Возвращает* CKR_RANDOM_SEED_NOT_SUPPORTED
C_GenerateRandom	*Получение случайного числа*	
PARALLEL FUNCTION MANAGEMENT FUNCTIONS/Функции параллельного управления		
C_GetFunctionStatus	*Функции управления параллельным выполнением крипто-операций*	
C_CancelFunction		

[10] *Функции C_WrapKey не поддерживаются СКЗИ «ESMART Token ГОСТ»*

[11] *Используется нестандартная реализация механизма CKM_GOST28147_KEY_WRAP, используемая для шифрования закрытого ключа ГОСТ Р 34.10, а не симметричного ключа ГОСТ 28147.*

7.2 Список дополнительных функций

PROPRIETARY FUNCTIONS/Проприетарные функции		
C_ISBC_CreateCSR	Создание запроса на выпуск сертификата	
C_ISBC_pkcs7Sign	Подпись данных в соответствии с PKCS#7	
C_ISBC_pkcs7Verify	Простая проверка подписи PKCS#7	
C_ISBC_CertVerify	Функция проверки сертификата в цепочке с возможностью проверки по списку отзыва	
C_ISBC_pkcs7VerifyEx	Расширенная проверка подписи, включающая в себя проверку цепочки сертификатов, наличие доверенного корневого сертификата и проверку по спискам отзыва	
C_ISBC_InitToken	Инициализация токена или карты с применением базовых параметров	
C_ISBC_InitTokenProfile	Инициализация токена или карты с применением профиля, т.е. набора расширенных параметров	
C_ISBC_GetProfile	Получение параметров профиля, заданного при инициализации функцией C_ISBC_GetProfile	
C_ISBC_GetCryptoProInfo	Получение информации о контейнерах CryptoPro, записанных на карту/токен	
C_ISBC_GetCryptoProCertificate	Получение информации о сертификате, хранящеся в контейнере CryptoPro, записанном на карту/токен	

C_ISBC_CheckSM	*Проверка возможности работать с проверкой Secure Messaging*[12]	*Возвращает* *0 – не поддерживается;* *1 – поддерживается, но не активирован;* *2 – поддерживается и активирован* *Для ESMART Token ГОСТ всегда возвращается 0.*
C_ISBC_ScribbleRead	*Чтение журнала событий на ESMART Token ГОСТ*	

*Коды стандартных ошибок перечислены в приложении B **Manifest constants** стандарта PKCS#11 (документ http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.pdf, стр. 142).*

Возможные ошибки, не описанные в стандарте, приведены в разделе 10.

[12] *Определение Secure Messaging по ГОСТ ИСО/МЭК 7816-4, статья 3.39*

8. Собственные функции

В разделе представлено подробное описание функций раздела PROPRIETARY FUNCTIONS

C_ISBC_CreateCSR

```
CK_RV C_ISBC_CreateCSR
(
      CK_SESSION_HANDLE       hSession,
      CK_OBJECT_HANDLE        publicKey,
      CK_CHAR_PTR             dn,
      CK_ULONG                dnLen,
      CK_BYTE_PTR             csr,
      CK_ULONG_PTR            cstLen,
      CK_OBJECT_HANDLE        privateKey,
      CK_CHAR_PTR             attrs,
      CK_ULONG                attrsLen,
      CK_CHAR_PTR             exts,
      CK_ULONG                extsLen
);
```

Возвращаемое значение:

CKR_OK – функция выполнена успешно.

Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Функция может возвращать коды ошибок CKR_VENDOR_TOO_MANY_ATTRIBUTES, CKR_VENDOR_CREATECSR_DN, CKR_VENDOR_CREATECSR_ATTR, CKR_VENDOR_CREATECSR_EXT (см. раздел 10), не описанные в стандарте.

Параметры:

hSession	*идентификатор сессии;*
publicKey	*идентификатор объекта типа* CKO_PUBLIC_KEY*, для которого создается запрос;*
dn	*Distinguished Name запроса. В параметре передается массив строк, где в первой строке располагается тип поля в текстовом формате или идентификатор объекта (CN - например). Во второй строке должно находиться значение поля в UTF8. Последующие поля передаются в следующих парах строк. Количество строк должно быть кратно двум;*
dnLen	*количество строк в массиве, на который ссылается параметр* dn*;*
csr	*указатель на память, куда будет помещен содержимое запроса. Память должна быть выделена заранее. Чтобы узнать необходимый размер памяти, необходимо вызвать функцию передав NULL в* csr*. Необходимый размер будет возвращен в параметре* cstLen*;*
cstLen	*размер буфера памяти под* csr*;*
privateKey	*идентификатор объекта типа* CKO_PRIVATE_KEY*, для которого создается запрос;*
attrs	*дополнительные атрибуты для включения в запрос. Формат аналогичен формату параметра* dn*;*
attrsLen	*количество строк в массиве, на который ссылается параметр* attrs*;*

`exts`	*расширения для включения в запрос. В параметре передается массив строк, где в первой строке располагается тип поля в текстовом формате или идентификатор объекта (keyUsage - например). Во второй строке должно находиться значение поля в UTF8. В третьей строке должен быть указано требование в данному расширению, 0 – не критическое, 1 – критическое. Последующие поля передаются в следующих строках. Количество строк должно быть кратно трем;*
`extsLen`	*количество строк в массиве, на который ссылается параметр* `exts`.

Особенности работы функции:

Параметры `dn`, `attr` *и* `ext` *должны быть представлены в виде непрерывного массива, в котором поля разделены нулевым символом* `'\0'`. *Например,* `dn` *необходимо передавать в виде* `"CN\0name\0C\0RU\0"`

Поддерживаются стандартные поля запроса, задаваемые в виде длинного или короткого имени:

Краткое имя	*Полное имя*	*OID*	*Описание*
Стандартные поля для параметров Distinguished name по RFC 5280			
CN	*commonName*	*2.5.4.3*	*Основное имя*
SN	*surname*	*2.5.4.4*	*Фамилия*
G	*givenName*	*2.5.4.42*	*Имя и отчество*
C	*countryName*	*2.5.4.6*	*Название страны, передается двумя заглавными буквами (RU/US/GB) согласно ГОСТ 7.67-2003 (ИСО 3166-1:1997)*
L	*localityName*	*2.5.4.7*	*Населенный пункт*
ST / S	*stateOrProvinceName*	*2.5.4.8*	*Регион*
-	*-*	*2.5.4.9*	*Адрес: улица и дом*
O	*organizationName*	*2.5.4.10*	*Название организации*
OU	*organizationalUnitName*	*2.5.4.11*	*Название структурной единицы (отдела)*
E	*emailAddress*	*1.2.840.113549.1.9.1*	*Адрес электронной почты*
Поля сертификата по ФЗ № 63-ФЗ и приказу № 795 «Об утверждении требований к форме квалифицированного сертификата ключа проверки электронной подписи»			
-	*-*	*1.2.643.100.1*	*ОГРН*
-	*-*	*1.2.643.100.3*	*СНИЛС*
-	*-*	*1.2.643.3.131.1.1*	*ИНН*

Для передачи параметров алгоритмов ГОСТ используются стандартные значения OID по RFC 4491 и методическим рекомендациям ТК26.

Для запроса на сертификат, содержащий идентификатор ключевой пары по алгоритму ГОСТ Р 34.10-2012 в 256-битном режиме, дополнительно необходимо передать OID 1.2.643.7.1.1.2.2 в качестве имени атрибута с любым значением атрибута. По умолчанию, если OID 1.2.643.7.1.1.2.2 не передан функции, запрос будет сформирован для ключевой пары по алгоритму ГОСТ Р 34.10-2001.

C_ISBC_pkcs7Sign

```
CK_RV C_ISBC_pkcs7Sign
 (
      CK_SESSION_HANDLE       hSession,
      CK_BYTE_PTR             pData,
      CK_ULONG                ulDataLen,
      CK_OBJECT_HANDLE        hSignCertificate,
      CK_BYTE_PTR             pOut,
      CK_ULONG_PTR            pulOutLen,
      CK_OBJECT_HANDLE        hPrivateKey,
      CK_OBJECT_HANDLE_PTR    hCertificates,
      CK_ULONG                ulCertificateLen,
      CK_ULONG                ulFlags
);
```

Возвращаемое значение:
CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Функция может возвращать не определенную стандартом ошибку CKR_VENDOR_CERT_EXPIRED *(см. раздел 10), если истек срок действия сертификата, который выбран для подписи.*

Параметры:

hSession	*идентификатор сессии;*
pData	*данные для подписи;*
ulDataLen	*размер данных для подписи;*
hSignCertificate	*идентификатор объекта типа* CKO_CERTIFICATE*, указывающий на сертификат создателя сообщения;*
pOut	*указатель на массив байт, в который передаются данные. Память должна быть выделена заранее. Чтобы узнать необходимый размер памяти необходимо вызвать функцию, передав NULL в* pOut*. Необходимый размер будет возвращен в параметре* pulOutLen*;*
pulOutLen	*указатель на длину созданного буфера;*
hPrivateKey	*идентификатор объекта типа* CKO_PRIVATE_KEY*, указывающий на закрытый ключ создателя сообщения;*
hCertificates	*указатель на массив сертификатов, которые следует добавить в сообщение;*
ulCertificateLen	*количество сертификатов в параметре* hCertificate*;*
ulFlags	*флаги. Переменная* CK_ULONG *может иметь следующие значения:* *0 – исходные данные, на которые ссылается указатель* pData*, сохраняются вместе с подписанным сообщением;* *0x40 – исходные данные, на которые ссылается указатель* pData*, не сохраняются вместе с подписанным сообщением.*

C_ISBC_pkcs7Verify

```
CK_RV C_ISBC_pkcs7Verify
 (
      CK_BYTE_PTR             pPkcs7,
      CK_ULONG                ulPkcs7Len,
      CK_BYTE_PTR             pData,
      CK_ULONG                ulDataLen
);
```

Возвращаемое значение:

CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Параметры:

pPkcs7	*указатель на байт-массив, содержащий объект данных, подписанный в формате PKCS#7;*
ulPkcs7Len	*размер объекта данных, на который ссылается указатель* pPkcs7*;*
pData	*указатель на массив, содержащий данные, если таковые отсутствуют непосредственно в самом объекте данных (если при создании подписи* ulFlags=0x40*);*
ulDataLen	*размер объекта данных, на который ссылается указатель* pData*.*

Функция C_ISBC_pkcs7Verify *используется только для базовой проверки математической верности электронной подписи. Для расширенной проверки электронной функции следует использовать функцию* C_ISBC_pkcs7VerifyEx*, включающую дополнительные возможности.*

C_ISBC_CertVerify

```
CK_RV ISBC_CertVerify
(
      CK_SESSION_HANDLE       hSession,
      CK_OBJECT_HANDLE        certificateToVerify,
      CK_OBJECT_HANDLE_PTR    trustedCertificates,
      CK_ULONG                trustedCertificatesLength,
      CK_OBJECT_HANDLE_PTR    certificateChain,
      CK_ULONG                certificateChainLength,
      CK_BYTE_PTR             *crls,
      CK_ULONG_PTR            crlsLengths,
      CK_ULONG                crlsLength
)
```

Возвращаемое значение:
`CKR_OK` – *функция выполнена успешно.*
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Функция может возвращать не определенную стандартом ошибку `CKR_VENDOR_VERIFY_ERR` *(см. раздел 10), если проверка сертификата завершилась с ошибкой.*

Параметры:

`hSession`	*идентификатор сессии;*
`certificateToVerify`	*идентификатор проверяемого сертификата;*
`trustedCertificates`	*указатель на массив доверенных сертификатов;*
`trustedCertificatesLength`	*размер массива;*
`certificateChain`	*указатель на массив сертификатов, не помеченных в качестве доверенных;*
`certificateChainLength`	*размер массива;*
`crls`	*указатель на массив указателей на списки отзыва;*
`crlsLengths`	*указатель на массив размеров списков отзыва;*
`crlsLength`	*размер массивов.*

Функция предназначена для проверки выбранного сертификата в доверенной цепочке. Для успешной проверки сертификата должны быть выполнены следующие условия:

- *проверяемый сертификат и каждый сертификат в цепочке должны иметь верную электронную подпись (быть математически верными);*
- *корневой сертификат иерархии и все промежуточные сертификаты в цепочке должны присутствовать на карте;*
- *опционально, функции могут быть переданы списки отзыва сертификатов в формате DER. Чтобы отключить проверку, вместо списков передается NULL.*

C_ISBC_pkcs7VerifyEx

```
CK_RV C_ISBC_pkcs7VerifyEx
(
      CK_SESSION_HANDLE       hSession,
      CK_BYTE_PTR             pPkcs7,
      CK_ULONG                ulPkcs7Len,
      CK_BYTE_PTR             pData,
      CK_ULONG                ulDataLen,
      CK_BYTE_PTR             *crls,
      CK_ULONG_PTR            crlsLengths,
      CK_ULONG                crlsLength,
      CK_BBOOL                ignoreCrlCheck
)
```

Возвращаемое значение:
CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

При неуспешной проверке функция возвращает ошибку CKR_VENDOR_VERIFY_ERR *(см. раздел 10) с указанием кода причины неуспешной проверки.*

Параметры:

hSession	*идентификатор сессии;*
pPkcs7	*указатель на байт-массив, содержащий объект данных, подписанный в формате PKCS#7;*
ulPkcs7Len	*размер объекта данных, на который ссылается указатель* pPkcs7*;*
pData	*указатель на массив, содержащий данные, если таковые отсутствуют непосредственно в самом объекте данных (если при создании подписи* ulFlags=0x40*);*
ulDataLen	*размер объекта данных, на который ссылается указатель* pData*;*
crls	*указатель на массив указателей на списки отзыва;*
crlsLengths	*указатель на массив размеров списков отзыва;*
crlsLength	*размер массивов;*
ignoreCrlCheck	*игнорировать проверку сертификата по списку отзыва.*

Условия успешной проверки подписи с помощью данной функции:

- *все сертификаты в цепочке должны быть математически верны (иметь верную электронную подпись);*
- *все сертификаты в цепочке должны иметь соответствующие расширения (например расширения для подписи данных для пользовательских сертфикатов, подписи CRL или других сертификатов для сертфиикатов УЦ);*
- *все сертификаты в цепочки должны быть действительны по сроку на момент выполнения подписи;*
- *корневой сертификат иерархии должен быть записан на карту с атрибутом CKA_TRUSTED;*
- *произведена успешная проверка всех сертификатов в цепочке по спискам отзыва.*

Для успешной проверки подписи списки отзыва должны удовлетворять следующим условиям:

- *все переданные списки отзыва должны быть математически верны (иметь верную электронную подпись);*
- *все списки отзыва не должны быть истекшими.*

Проверка по списку отзыва сертификатов повышает уровень доверия. Текущие списки отзыва сертификатов должны регулярно обновляться на ПЭВМ, на которых проводится проверка подписи. Отключать возможность проверки подписи допускается только для проверки функционирования системы.

Сертификаты и списки отзыва CRL должны быть переданы в функцию в кодировке DER.

C_ISBC_InitToken

```
CK_RV C_ISBC_InitToken
 (
      CK_SLOT_ID              slotID,
      CK_UTF8CHAR_PTR         pPin,
      CK_ULONG                ulPinLen,
      CK_UTF8CHAR_PTR         pLabel,
      CK_BYTE                 bSoPinRc,
      CK_BYTE                 bUserPinRc
);
```

Возвращаемое значение:

`CKR_OK` – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Параметры:

`slotID`	*ID слота/считывателя смарт-карт;*
`pPin`	*значение SO PIN;*
`ulPinLen`	*длина `pPin`;*
`pLabel`	*метка токена;*
`bSoPinRc`	*число попыток ввода неверного SO PIN подряд. Фиксированное значение, равное 10 попыткам, задается при первоначальной инициализации криптографического чипа.* *Ограничение в 10 попыток нельзя изменить при инициализации токена/карты функциями PKCS11. Переданное функции значение игнорируется;*
`bUserPinRc`	*число попыток ввода неверного User PIN подряд. Значение должно быть в диапазоне >0 и <= 254. Таким образом, максимальное значение количества попыток равно 254.* *Полностью снять ограничение на количество попыток ввода неверного ПИН-кода пользователя подряд невозможно.*

C_ISBC_InitTokenProfile

```
CK_RV C_ISBC_InitTokenProfile
(
      CK_SLOT_ID              slotID,
      CK_UTF8CHAR_PTR         pPin,
      CK_ULONG                ulPinLen,
      CK_UTF8CHAR_PTR         pLabel,
      CK_BYTE                 bProfileId,
      CK_BYTE                 bSoPinRc,
      CK_BYTE                 bUserPinRc,
      CK_BYTE                 bDataFileSize,
      CK_BYTE                 bDataFileCount,
      CK_BYTE                 bCertFileSize,
      CK_BYTE                 bCertFileCount,
      CK_BYTE                 bPubFileSize,
      CK_BYTE                 bPubFileCount,
      CK_BYTE                 bPrFileSize,
      CK_BYTE                 bPrFileCount,
      CK_BYTE                 bSKFileSize,
      CK_BYTE                 bSKFileCount,
      CK_BYTE                 bSM,
      CK_BYTE                 bConf
)
```

Возвращаемое значение:

`CKR_OK` – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Параметры:

`slotID`	*ID слота/считывателя смарт-карт;*
`pPin`	*значение SO PIN;*
`ulPinLen`	*длина `pPin`;*
`pLabel`	*метка токена;*
`bProfileId`	*идентификатор профиля;*
`bSoPinRc`	*число попыток ввода неверного SO PIN подряд. Фиксированное значение, равное 10 попыткам, задается при первоначальной инициализации криптографического чипа.* *Ограничение в 10 попыток нельзя изменить при инициализации токена/карты функциями PKCS11. Переданное функции значение игнорируется;*
`bUserPinRc`	*число попыток ввода неверного User PIN подряд. Значение должно быть в диапазоне >0 и <= 254. Таким образом, максимальное значение количества попыток равно 254.* *Полностью снять ограничение на количество попыток ввода неверного ПИН-кода пользователя подряд невозможно.*
`bSoPinRc`	*число попыток ввода SO PIN, значение должны быть >0 и <= 0x0F. Значения 0x0F означает, что счетчик отключен (если карта/токен это поддерживает);*
`bUserPinRc`	*число попыток ввода User PIN, значение должны быть >0 и <= 0x0F. Значения 0x0F означает, что счетчик отключен (если карта/токен это поддерживает);*

bDataFileSize	*размер блока памяти, отведенный под один объект данных* CKO_DATA. *Длина одного блока составляет 256 байт. Переданное значение игнорируется, размер блока определяется динамически при записи конкретного объекта;*
bDataFileCount	*изначальное количество объектов данных при задании параметров файловой системы;*
bCertFileSize	*размер блока памяти, отведенный под один объект сертификатов* CKO_CERTIFICATE. *Длина одного блока составляет 256 байт. Переданное значение игнорируется, размер блока определяется динамически при записи конкретного объекта;*
bCertFileCount	*изначальное количество объектов сертификатов при задании параметров файловой системы;*
bPubFileSize	*размер блока памяти, отведенный под один объект открытого ключа* CKO_PUBLIC_KEY. *Длина одного блока составляет 256 байт. Переданное значение игнорируется, размер блока определяется динамически при записи конкретного объекта;*
bPubFileCount	*изначальное количество объектов открытых ключей при задании параметров файловой системы;*
bPrFileSize	*размер блока памяти, отведенный под один объект закрытого ключа* CKO_PRIVATE_KEY. *Длина одного блока составляет 256 байт. Переданное значение игнорируется, размер блока определяется динамически при записи конкретного объекта;*
bPrFileCount	*изначальное количество объектов закрытых ключей при задании параметров файловой системы;*
bSKFileSize	*размер блока памяти, отведенный под один объект симметричного ключа* CKO_SECRET_KEY. *Один блок занимает 256 байт. Переданное значение игнорируется, размер блока определяется динамически при записи конкретного объекта;*
bSKFileCount	*изначальное количество объектов симметричных ключей при задании параметров файловой системы;*
bSM	*активация опции Secure Messaging. Всегда следует передавать 0, т.к. данная опция не поддерживается;*
bConf	*Дополнительные параметры профиля, например, активация работы с биометрией (не поддерживается). Рекомендуется устанавливать значение 0. Если при создании объекта значение не было задано, используется значение по умолчанию 0x04.*

Функция предназначена для разметки файловой системы при инициализации. В отличие от функции C_ISBC_InitToken *в данной функции присутствует возможность задать дополнительные параметры разметки файловой системы.*

В соответствии с особенностями работой файловой системы смарт-карт и токенов СКЗИ «ESMART Token ГОСТ» параметры объектов, заданные при инициализации, будут динамически изменены при создании новых объектов.

Параметры, соответствующие размеру блока, при инициализации могут быть переданы любые. При фактическом создании объекта его размер будет задан динамически. Параметры, определяющие количество создаваемых объектов, задают только первоначальное значение. Например, при инициализации

передано значение `bCertFileCount` *равное 3. При попытке создания 4-го объекта типа «сертификат» автоматически будет выделено новое адресное пространство в памяти для создания новых объектов.*

В качестве значения параметров `bSM` *и* `bConf` *следует передавать 0. Можно передавать и любое другое значение, но оно будет игнорироваться.*

C_ISBC_GetProfile

```
CK_RV C_ISBC_GetProfile
(
      CK_SLOT_ID              slotID,
      CK_BYTE                 *pbVerMsb,
      CK_BYTE                 *pbVerLsb,
      CK_BYTE                 *pbProfileId,
      CK_BYTE                 *pbSoPinRc,
      CK_BYTE                 *pbUserPinRc,
      CK_BYTE                 *pbDataFileSize,
      CK_BYTE                 *pbDataFileCount,
      CK_BYTE                 *pbCertFileSize,
      CK_BYTE                 *pbCertFileCount,
      CK_BYTE                 *pbPubFileSize,
      CK_BYTE                 *pbPubFileCount,
      CK_BYTE                 *pbPrFileSize,
      CK_BYTE                 *pbPrFileCount,
      CK_BYTE                 *pbSKFileSize,
      CK_BYTE                 *pbSKFileCount,
      CK_BYTE                 *pbSM,
      CK_BYTE                 *pbConf
)
```

Параметры:

`slotID`	*ID слота/считывателя смарт-карт;*
`pbVerMsb`	*старший байт номера версии разметки токена, возвращает 1;*
`pbVerLsb`	*старший байт номера версии разметки токена, возвращает 1;*
`pbProfileId`	*идентификатор профиля, возвращается значение, заданное при инициализации с данным профилем;*
`pbSoPinRc`	*число попыток ввода SO PIN, значение должны быть >0 и <= 0x0F. Значения 0x0F означает, что счетчик отключен (если карта/токен это поддерживает);*
`pbUserPinRc`	*число попыток ввода User PIN, значение должны быть >0 и <= 0x0F. Значения 0x0F означает, что счетчик отключен (если карта/токен это поддерживает);*
`pbDataFileSize`	*размер блока памяти, отведенный под один объект данных* `CKO_DATA`. *Длина одного блока составляет 256 байт. Переданное значение игнорируется, размер блока определяется динамически при записи конкретного объекта;*
`pbDataFileCount`	*изначальное количество объектов симметричных ключей при задании параметров файловой системы;*
`pbPubFileSize`	*размер блока памяти, отведенный под один объект данных* `CKO_DATA`. *Длина одного блока составляет 256 байт. Возвращаемое значение актуально только для информации, размер блока определяется динамически при записи конкретного объекта;*
`pbPubFileCount`	*изначальное количество объектов симметричных ключей при задании параметров файловой системы;*
`pbPrFileSize`	*размер блока памяти, отведенный под один объект данных* `CKO_DATA`. *Длина одного блока составляет 256 байт. Возвращаемое значение актуально только для информации, размер блока определяется динамически при записи конкретного объекта;*
`pbPrFileCount`	*изначальное количество объектов симметричных ключей при задании параметров файловой системы;*

pbSKFileSize	*размер блока памяти, отведенный под один объект данных* CKO_DATA. *Длина одного блока составляет 256 байт. Возвращаемое значение актуально только для информации, размер блока определяется динамически при записи конкретного объекта;*
pbSKFileCount	*изначальное количество объектов симметричных ключей при задании параметров файловой системы;*
pbSM	*Использование Secure Messaging, всегда возвращается 0;*
pbConf	*использование биометрии, возвращается то значение, которое было задано при создании профиля функцией* C_ISBC_InitTokenProfile. *По умолчанию функция* C_ISBC_InitTokenProfile *устанавливает значение 0x04.*

C_ISBC_GetCryptoProInfo

```
CK_RV C_ISBC_GetCryptoProInfo
(
      CK_SLOT_ID              slotID,
      CK_ULONG                *pulCount,
      CK_BYTE                 *pbNames,
      CK_ULONG                *pulNamesLen
);
```

Возвращаемое значение:
CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Параметры:

slotID	*ID слота/считывателя смарт-карт;*
pulCount	*количество контейнеров;*
pbNames	*имена контейнеров. Память должна быть выделена заранее. Чтобы узнать необходимый размер памяти необходимо вызвать функцию передав NULL в* pbNames*. Необходимый размер будет возвращен в параметре* pulNamesLen*;*
pulNamesLen	*размер буфера* pbNames*;*

Работа с функцией выполняется в 2 этапа.
На первом этапе определяется требуемый объем выделяемой памяти. Для этого функция вызывается первый раз и в параметре pbNames *передается NULL. Функция вернет размер массива, содержащего TLV-структуры с именами контейнеров КриптоПРО, в параметре* pulNamesLen*.*

На втором этапе нужно выделить память под этот массив, и вызвать функцию повторно, передав указатель на него в параметре pbNames*.*

TLV-структуры с именами контейнеров имеют следующий формат:
Формат pbNames*: последовательно тегов 0x30, содержащих тег 0x16 с именем контейнера в кодировке ISO 646 (IA5) (преобразуется к CP1251).*

C_ISBC_GetCryptoProCertificate

```
CK_RV C_ISBC_GetCryptoProCertificate
(
      CK_SLOT_ID              slotID,
      CK_BYTE                 *pbCertName,
      CK_ULONG                ulNameLen,
      CK_BYTE                 *pbCert,
      CK_ULONG                *pulCertLen
);
```

Возвращаемое значение:

CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Параметры:

`slotID`	*ID слота/считывателя смарт-карт;*
`pbCertName`	*Имя контейнера криптопро в кодировке CP1251, для которого запрашивается чтение сертификата;*
`ulNameLen`	*Размер массива* `pbCertName`*;*
`pbCert`	*Массив, содержащий сертификат КриптоПРО из контейнера с именем* `pbCertName`*;*
`pulCertLen`	*Размер буфера* `pbCert`*;*

Для чтения сертификата из контейнера необходимо получить имя контейнера с помощью функции `C_ISBC_GetCryptoProInfo`.

Работа с функцией выполняется в 2 этапа.

Сперва следует вызвать эту функцию, передав в параметре `pbCertName` *имя контейнера КриптоПРО, полученного на предыдущем шаге в кодировке CP1251, размер этого массива в параметре* `ulNameLen` *и NULL в параметре* `pbCert`.
Функция вернет размер массива, содержащего сертификат КриптоПРО, в параметре `pulCertLen`.

Вторым шагом нужно выделить память под этот массив, и вызвать функцию повторно, передав указатель на него в параметре `pbCert`.

C_ISBC_CheckSM

```
CK_RV C_ISBC_CheckSM
(
      CK_SLOT_ID              slotID,
      CK_BYTE                 *pbRes,


);
```

Возвращаемое значение:
CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Параметры:

slotID	*ID слота/считывателя смарт-карт;*
pbRes	*Указатель на возвращаемое значение:* *0 – не поддерживается;* *1 – поддерживается, но не активирован;* *2 – поддерживается и активирован.*

Для смарт-карт и токенов СКЗИ «ESMART Token ГОСТ» опция Secure Messaging не поддерживается. Функция всегда возвращает 0.

C_ISBC_ScribbleRead

```
CK_RV C_ISBC_ScribbleRead
(
      CK_SESSION_HANDLE       hSession,
      CK_BYTE                 type,
      CK_BYTE_PTR             pData,
      CK_ULONG_PTR            pulDataLen
);
```

Возвращаемое значение:
`CKR_OK` – *функция выполнена успешно.*
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Функция может возвращать код ошибки CKR_VENDOR_SCRIB_NOT_FOUND (см. раздел 10), если файл журнала событий отсутствует.

Параметры:

`type`	*должно равняться 0;*
`pData`	*указатель на буфер. NULL если нужно узнать размер;*
`pulDataLen`	*размер буфера*

Формат каждой записи - 6 байт. Первые 4 байта представляют дату записи в секундах, начиная с 01.01.2016 0:00:00 UTC. Следующий байт передает код события. Последний байт зарезервирован (RFU).

Код события:

0x01 – Первичная персонализация
0x02 – Инициализация функцией C_InitToken
0x03 – Смена ПИН пользователя
0x04 – Смена ПИН администратора
0x05 – Не удачная попытка авторизации по ПИН пользователя
0x06 – Не удачная попытка авторизации по ПИН администратора
Коды 0x03 - 0x06 могут комбинироваться с флагом 0x80, если событие связано с минидрайвером.
Коды 0x05 - 0x06 могут комбинироваться с флагом 0x40, если источник события неизвестен.

9. Реализация с особенностями

C_UnwrapKey

```
CK_RV C_UnwrapKey
(
CK_SESSION_HANDLE            hSession,
  CK_MECHANISM_PTR           pMechanism,
  CK_OBJECT_HANDLE           hUnwrappingKey,
  CK_BYTE_PTR                pWrappedKey,
  CK_ULONG                   ulWrappedKeyLen,
  CK_ATTRIBUTE_PTR           pTemplate,
  CK_ULONG                   ulAttributeCount,
  CK_OBJECT_HANDLE_PTR       phKey
);
```

Возвращаемое значение:

CKR_OK – функция выполнена успешно.
Прочие возможные значения реализованы так, как они описаны в стандарте PKCS#11 (документ pkcs11-base-v2.40-os стр. 142).

Параметры:

hSession	*дескриптор сессии;*
pMechanism	*поддерживается только механизм CKM_GOST28147_KEY_WRAP;*
hUnwrappingKey	*указатель на ключ расшифрования;*
pWrappedKey	*в функцию C_UnwrapKey передается зашифрованная на hUnwrappingKey (CKK_GOST28147) структура PKCS #8 для PrivateKeyInfo ASN.1, представленная согласно дополнения ТК26;*
ulWrappedKeyLen	*длина данных;*
pTemplate	*шаблон атрибутов;*
ulAttributeCount	*количество атрибутов в шаблоне;*
phKey	*показывает, куда передать указатель на восстановленный (расшифрованный) ключ.*

Используется нестандартная реализация механизма CKM_GOST28147_KEY_WRAP, используемая для шифрования только закрытого ключа ГОСТ Р 34.10, а не симметричного ключа ГОСТ 28147.

В функцию C_UnwrapKey передается зашифрованная на hUnwrappingKey (CKK_GOST28147) структура PKCS#8 для PrivateKeyInfo ASN.1, представленная согласно дополнения ТК26.

10. Обработка ошибок

*Коды ошибок перечислены в приложении B **Manifest constants** стандарта PKCS#11 (документ http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.pdf, стр. 142).*

Дополнительно могут возвращаться коды ошибок, не входящие в стандарт PKCS#11:

```
CKR_VENDOR_TOO_MANY_ATTRIBUTES      ( CKR_VENDOR_DEFINED | 0x0001 )
```

Ошибка может возвращаться при создании объектов любого типа, если передано слишком много атрибутов.

```
CKR_VENDOR_CREATECSR_DN             ( CKR_VENDOR_DEFINED | 0x0301 )
CKR_VENDOR_CREATECSR_ATTR           ( CKR_VENDOR_DEFINED | 0x0302 )
CKR_VENDOR_CREATECSR_EXT            ( CKR_VENDOR_DEFINED | 0x0303 )
```

для индикации проблем с параметрами `dn, attrs, exts C_ISBC_CreateCSR`.

```
CKR_VENDOR_SCRIB_NOT_FOUND          ( CKR_VENDOR_DEFINED | 0x0101 )
```

в случае, когда функция `C_ISBC_ScribbleRead` *не находит файл истории.*

```
CKR_VENDOR_CERT_EXPIRED             ( CKR_VENDOR_DEFINED | 0x0600 )
```

Срок действия сертификата, который предполагается использовать для подписи, истек. Может возвращаться функцией `C_ISBC_pkcs7Sign`.

```
CKR_VENDOR_VERIFY_ERR               ( CKR_VENDOR_DEFINED | 0x0500 )
```

Ошибка, возвращаемая функциями `C_ISBC_CertVerify` *и* `C_ISBC_pkcs7VerifyEx`.

Для функции проверки цепочки сертификатов `C_ISBC_CertVerify` *всегда возвращается только общий код ошибки.*

Для функции проверки электронной подписи `C_ISBC_pkcs7VerifyEx`*, финальный код ошибки является конкатенацией кода* `CKR_VENDOR_VERIFY_ERR` *и соответствующей специфической ошибки, приведенной в разделе 10.1.*

10.1 Специфические коды, возвращаемые функцией C_ISBC_pkcs7VerifyEx

X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT	*0x02*
Не найден сертификат издателя (корневой или промежуточный сертификат в цепочке)	
X509_V_ERR_UNABLE_TO_GET_CRL	*0x03*
Не найден список отзыва (CRL) для сертификата	
X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE	*0x04*
Невозможно расшифровать подпись сертификата. Это значит, что невозможно получить само значение подписи, а не то, что это значение не соответствует ожидаемому. Применяется только для алгоритма RSA	
X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE	*0x05*
Невозможно расшифровать подпись CRL. Это значит, что невозможно получить само значение подписи, а не то, что это значение не соответствует ожидаемому. Не используется	

X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY	*0x06*
Невозможно декодировать открытый ключ издателя сертификата. Невозможно прочитать параметр SubjectPublicKeyInfo	
X509_V_ERR_CERT_SIGNATURE_FAILURE	*0x07*
Проверка подписи сертификата завершилась с ошибкой	
X509_V_ERR_CRL_SIGNATURE_FAILURE	*0x08*
Проверка подписи CRL завершилась с ошибкой	
X509_V_ERR_CERT_NOT_YET_VALID	*0x09*
Сертификат не действителен: срок действия сертификата не вступил в силу	
X509_V_ERR_CERT_HAS_EXPIRED	*0x0A*
Сертификат не действителен: срок действия сертификата истек	
X509_V_ERR_CRL_NOT_YET_VALID	*0x0B*
Список отзыва недействителен: срок действия списка (CRL) не вступил в силу	
X509_V_ERR_CRL_HAS_EXPIRED	*0x0C*
Список отзыва недействителен: срок действия списка (CRL) истек	
X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD	*0x0D*
Недопустимое значение в поле сертификата notBefore, которое определяет начальный срок действия сертификата	
X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD	*0x0E*
Недопустимое значение в поле сертификата notAfter, которое определяет срок истечения сертификата	
X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD	*0x0F*
Недопустимое значение в поле lastUpdate, которое определяет дату и время последнего обновления списка отзыва	
X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD	*0x10*
Недопустимое значение в поле nextUpdate, которое определяет дату и время следующего обновления	
X509_V_ERR_OUT_OF_MEM	*0x11*
Возникла ошибка при выделении памяти. Код ошибки предусмотрен, но фактически не возвращается	
X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT	*0x12*
Проверка завершилась ошибкой, т.к. переданный сертификат является самоподписанным и при этом не найден в списке доверенных сертификатов на карте/токене (имеющих атрибут CKA_TRUSTED)	
X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN	*0x13*
При проверке цепочки корневой сертификат не обнаружен среди доверенных сертификатов на карте/токене	
X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY	*0x14*
Сертификат издателя недоверенный. Вероятно, отсутствуют один или несколько промежуточных сертификатов в цепочке	
X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE	*0x15*
Подпись не может быть проверена, т.к. в цепочке присутствует только один сертификат, и он не является самоподписанным (корневым)	

X509_V_ERR_CERT_CHAIN_TOO_LONG	*0x16*
Превышена максимальная длина цепочки. Ошибка предусмотрена, но фактически не возвращается	
X509_V_ERR_CERT_REVOKED	*0x17*
Сертификат отозван	
X509_V_ERR_INVALID_CA	*0x18*
Недопустимый сертификат удостоверяющего центра. Возможно, в сертификате нет необходимых расширений (ext) для запрашиваемого способа использования сертификата	
X509_V_ERR_PATH_LENGTH_EXCEEDED	*0x19*
Превышено допустимое значение параметра длины пути basicConstraints pathlength	
X509_V_ERR_INVALID_PURPOSE	*0x1A*
Проверяемый сертификат не может использоваться для указанной операции	
X509_V_ERR_CERT_UNTRUSTED	*0x1B*
Сертификат корневого центра сертификации не помечен как доверенный для указанной операции	
X509_V_ERR_CERT_REJECTED	*0x1C*
Сертификат корневого центра сертификации имеет запрет на использование для данной операции	
X509_V_ERR_SUBJECT_ISSUER_MISMATCH	*0x1D*
Текущий сертификат издателя не прошел проверку соответствия имени издателя (subject) в данном сертификате с именем издателя в проверяемом сертификате, хранящемся локально. Не считается ошибкой проверки	
X509_V_ERR_AKID_SKID_MISMATCH	*0x1E*
Текущий сертификат издателя не прошел проверку соответствия идентификатора ключа в данном сертификате с идентификатором ключа издателя в проверяемом сертификате, хранящемся локально. Не считается ошибкой проверки	
X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH	*0x1F*
Текущий сертификат издателя не прошел проверку соответствия серийного номера и идентификатора ключа в данном сертификате с серийным номером и идентификатором ключа издателя в проверяемом сертификате, хранящемся локально. Не считается ошибкой проверки	
X509_V_ERR_KEYUSAGE_NO_CERTSIGN	*0x20*
Сертификат издателя был отклонен, т.к. он не содержит значений в разделе расширений keyUsage extension для подписи сертификата. Не считается ошибкой проверки	
X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER	*0x21*
Невозможно получить сертификат издателя списка отзывов	
X509_V_ERR_KEYUSAGE_NO_CERTSIGN	*0x20*
Сертификат издателя был отклонен, т.к. он не содержит значений в разделе расширений keyUsage extension для подписи сертификата. Не считается ошибкой проверки	
X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER	*0x21*
Невозможно получить сертификат издателя списка отзывов	
X509_V_ERR_UNHANDLED_CRITICAL_EXTENSION	*0x22*
Не может быть обработано одно из критических расширений	
X509_V_ERR_KEYUSAGE_NO_CRL_SIGN	*0x23*
В расширениях сертификата отсутствует разрешением для подписи списка отзывов	

X509_V_ERR_UNHANDLED_CRITICAL_CRL_EXTENSION	*0x24*
Не может быть обработано одно из критических расширений списка отзывов	
X509_V_ERR_INVALID_NON_CA	*0x25*
Попытка использовать в качестве сертификата издателя сертификат, не являющийся сертификатом издателя (не имеющий требуемых расширений)	
X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED	*0x26*
Превышена максимально допустимая длина цепочки прокси-серверов. Используется только в соответствии с RFC 3820	
X509_V_ERR_KEYUSAGE_NO_DIGITAL_SIGNATURE	*0x27*
Сертификат не имеет привелегии подписи Digital Signature *в расширении Enhanced Key Usage (EKU)*	
X509_V_ERR_PROXY_CERTIFICATES_NOT_ALLOWED	*0x28*
Сертификаты прокси-серверов не разрешены	
X509_V_ERR_INVALID_EXTENSION	*0x29*
Расширение сертификата содержит недопустимое значение (например, неверно закодировано) или одновременно добавлены взаимоисключающие расширения	
X509_V_ERR_INVALID_POLICY_EXTENSION	*0x2A*
Расширение политики сертификата содержит недопустимое значение (например, неверно закодировано) или одновременно добавлены взаимоисключающие расширения	
X509_V_ERR_NO_EXPLICIT_POLICY	*0x2B*
Выставлен флаг требовать принудительной проверки политик сертификата, но политики отсутствует	
X509_V_ERR_DIFFERENT_CRL_SCOPE	*0x2C*
В переданном списке отзывов недопустимый набор сертификатов	
X509_V_ERR_UNSUPPORTED_EXTENSION_FEATURE	*0x2D*
Не поддерживается свойство расширения сертификата. Ошибка предусмотрена, но фактически никогда не возвращается	
X509_V_ERR_PERMITTED_VIOLATION	*0x2F*
Нарушение ограничений форматов для имен в разрешенной части	
X509_V_ERR_EXCLUDED_VIOLATION	*0x30*
Нарушение ограничений форматов для имен в исключенной части	
X509_V_ERR_SUBTREE_MINMAX	*0x31*
Ограничение для имен включает поля MIN/MAX: не поддерживается	
X509_V_ERR_APPLICATION_VERIFICATION	*0x32*
Ошибка, которую возвращает стороннее приложение при проверке. Данный код может быть возвращен, только если явно задан самим приложением.	
X509_V_ERR_UNSUPPORTED_CONSTRAINT_TYPE	*0x33*
Неподдерживаемое ограничение для данного типа имени. Поддерживаются только проверка следующих типов имен: имя директории, имя DNS, email и URI.	
X509_V_ERR_UNSUPPORTED_CONSTRAINT_SYNTAX	*0x34*
Формат ограничения имени не может быть распознан, например, при попытке проверки формата, не указанного в RFC3280 для адреса электронной почты	
X509_V_ERR_UNSUPPORTED_NAME_SYNTAX	*0x35*
Неподдерживаемый синтаксис имени	

X509_V_ERR_CRL_PATH_VALIDATION_ERROR	*0x36*
Ошибка проверки пути к списку отзыва сертификата	