ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ
(РОССТАНДАРТ)

Технический комитет 026

«Криптографическая защита информации»

Информационная технология

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Расширение PKCS#11 для использования российских стандартов ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012, ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015

(проект)

Москва

2015

Содержание

1. Аннотация

Данный документ определяет расширение спецификаций PKCS#11 для использования криптографических алгоритмов **ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012, ГОСТ Р 34.12-2015** и **ГОСТ Р 34.13-2015**, а также алгоритмов, построенных на их основе, а также использования ключей, сертификатов, параметров алгоритмов и других объектов, предназначенных для работы с этими стандартами.

2. Список ссылок

В настоящем документе использованы ссылки на следующие стандарты и рекомендации:

- **ГОСТ Р 34.10-2012** — «Информационные технологии. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи», ГОСТ Р 34.10-2012, Национальный стандарт Российской Федерации, Федеральное агентство по техническому регулированию и метрологии, Стандартинформ, 2012.
- **ГОСТ Р 34.11-2012** — «Информационные технологии. Криптографическая защита информации. Функция хэширования», ГОСТ Р 34.11-2012, Национальный стандарт Российской Федерации, Федеральное агентство по техническому регулированию и метрологии, Стандартинформ, 2012.
- **ГОСТ Р 34.12-2015 —** «Информационная технология. Криптографическая защита информации. Блочные шифры», ГОСТ Р 34.12-2015, Национальный стандарт Российской Федерации, Федеральное агентство по техническому регулированию и метрологии, Стандартинформ, 2015.
- **ГОСТ Р 34.13-2015 —** «Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров», ГОСТ Р 34.13-2015, Национальный стандарт Российской Федерации, Федеральное агентство по техническому регулированию и метрологии, Стандартинформ, 2015.
- **ТК26АЛГ** — Федеральное агентство по техническому регулированию и метрологии (РОС-СТАНДАРТ), Технический комитет №26, «Рекомендации по стандартизации. криптографическим алгоритмам, сопутствующим применению стандартов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012.
- **ТК26ЭК** — Федеральное агентство по техническому регулированию и метрологии (РОС-СТАНДАРТ), Технический комитет №26, «Рекомендации по заданию параметров эллиптических кривых в соответствии с ГОСТ Р 34.10-2012», Москва, 2014.
- **ТК26ЭДВ —** Федеральное агентство по техническому регулированию и метрологии (РОС-СТАНДАРТ), Технический комитет №26, «Рекомендации по стандартизации. Задание параметров скрученных эллиптических кривых Эдвардса в соответствии с ГОСТ Р 34.10-2012», Москва, 2013
- **RFC 4357** — В. Попов, И. Курепкин, С. Леонтьев, «Дополнительные алгоритмы шифрования для использования с алгоритмами по ГОСТ 28147-89, ГОСТ Р 34.10-94, ГОСТ Р 34.10-2001 и ГОСТ Р 34.11-94» (Popov V., Kurepkin I. and S. Leontiev,

Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms, IETF RFC 4357, January 2006).

3. Замечание об использовании численных идентификаторов

До момента включения данного дополнения в официальный профиль PKCS#11 и назначения «стандартных» значений для всех приведенных здесь определений численные значения для них выбираются в соответствии со следующими правилами:

1) признаком нестандартного значения (определяемого производителем) является взведенный старший бит (0x80000000);
2) в каждом из самостоятельных «пространств имен» определений значения выбираются произвольно с учетом уникальной базы и идентификатора производителя.
В качестве идентификатора производителя выбрано следующее значение:

```
#define NSSCK_VENDOR_PKCS11_RU_TEAM 0xd4321000 //0x80000000|0x54321000
#define CK_VENDOR_PKCS11_RU_TEAM_TC26 NSSCK_VENDOR_PKCS11_RU_TEAM
```

4. Тип ключа алгоритма ГОСТ Р 34.10-2012

Данное расширение PKCS#11 вводит определение в пространстве значений типов ключей:

```
#define CKK_GOSTR3410_512 (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x003)
```

Значение CKK_GOSTR3410_512 может являться значением атрибута CKA_KEY_TYPE для CKO_PUBLIC_KEY и CKO_PRIVATE_KEY.

Ключ с типом CKK_GOSTR3410_512 предназначен для использования с алгоритмом **ГОСТ Р 34.10-2012** с длиной закрытого ключа 512 бит.

Такой ключ может использоваться с механизмами

```
CKM_GOSTR3410_512_KEY_PAIR_GEN
CKM_GOSTR3410_512
CKM_GOSTR3410_2012_DERIVE
```

5. Тип ключа алгоритма ГОСТ Р 34.12-2015

Вводится тип ключа:

```
#define CKK_KUZNECHIK (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x004)
```

Значение CKK_KUZNECHIK может являться значением атрибута CKA_KEY_TYPE для CKO_SECRET_KEY.

Ключ с типом CKK_KUZNECHIK предназначен для использования с алгоритмами **ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015** с длиной закрытого ключа 256 бит.

6. Наборы параметров

В качестве параметров алгоритма **ГОСТ Р 34.10-2012** допускается использование следующих параметров:

Для длины закрытого ключа 256 бит:

- Параметры подписи по умолчанию с ключом 256 бит id-GostR3410-2001-CryptoPro-A-ParamSet. OID "1.2.643.2.2.35.1". ASN представление {0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x01}. Определён в **RFC 4357**.
- Параметры id-GostR3410-2001-CryptoPro-B-ParamSet.OID "1.2.643.2.2.35.2". ASN представление {0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x02}. Определён в **RFC 4357**.
- Параметры id-GostR3410-2001-CryptoPro-C-ParamSet. OID "1.2.643.2.2.35.3". ASN представление {0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x23, 0x03}. Определён в **RFC 4357**.
- Параметры обмена по умолчанию с ключом 256 бит id-GostR3410-2001-CryptoPro-XchA-ParamSet. OID "1.2.643.2.2.36.0". ASN представление {0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x24, 0x00}. Определён в **RFC 4357**.
- Параметры id-GostR3410-2001-CryptoPro-XchB-ParamSet. OID "1.2.643.2.2.36.1". ASN представление {0x06, 0x07, 0x2A, 0x85, 0x03, 0x02, 0x02, 0x24, 0x01}. Определён в **RFC 4357**.
- Параметры с ключом 256 для скрученных эллиптических кривых Эдвардса id-tc26-gost-3410-2012-256-paramSetA. OID "1.2.643.7.1.2.1.1.1". ASN представление { 0x06, 0x09, 0x2A, 0x85, 0x03, 0x07, 0x01, 0x02, 0x01, 0x01, 0x01 }. Определён в **ТК26ЭДВ**.

Для длины закрытого ключа 512 бит:

- Параметры по умолчанию с ключом 512 бит id-tc26-gost-3410-2012-512-paramSetA. OID "1.2.643.7.1.2.1.2.1". ASN представление { 0x06, 0x09, 0x2A, 0x85, 0x03, 0x07, 0x01, 0x02, 0x01, 0x02, 0x01 }. Определён в **ТК26ЭК**.
- Рабочие параметры с ключом 512 id-tc26-gost-3410-2012-512-paramSetB. OID "1.2.643.7.1.2.1.2.2". ASN представление { 0x06, 0x09, 0x2A, 0x85, 0x03, 0x07, 0x01, 0x02, 0x01, 0x02, 0x02 }. Определён в **ТК26ЭК**.
- Параметры с ключом 512 для скрученных эллиптических кривых Эдвардса id-tc26-gost-3410-2012-512-paramSetC. OID "1.2.643.7.1.2.1.2.3". ASN представление { 0x06, 0x09, 0x2A, 0x85, 0x03, 0x07, 0x01, 0x02, 0x01, 0x02, 0x03 }. Определён в **ТК26ЭДВ**.

Набор параметров, идентифицируемый OID-ом 1.2.643.7.1.2.1.2.1, является набором по умолчанию для закрытого ключа длины 512 бит. Если в параметрах механизма явно не заданы параметры, используется этот набор.

В объектах ключей и шаблонах, которые передаются в функции, в качестве значений атрибутов CKA_GOSTR3410_PARAMS и CKA_GOSTR3410_512PARAMS, эллиптические параметры представляются в виде ASN закодированного OID.

7. Механизмы хэширования

Список механизмов хэширования расширяется для использования нового российского стандарта ГОСТ Р 34.11-2012. Для двух функций хэширования, вводятся два новых механизма.

Стандарт ГОСТ Р 34.11-2012 определяет две функции хэширования с длинами значений 256 и 512 бит. Для использования этих функций в PKCS#11 вводятся два механизма

```
#define CKM_GOSTR3411_2012_256 (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x012)
#define CKM_GOSTR3411_2012_512 (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x013)
```

с длинами значений 32 и 64 байта соответственно. Эти механизмы не используют параметры.

Используется в C_DigestInit.

8. Механизмы hmac

Определяются две функции hmac, базирующиеся на основании функций хэширования ГОСТ Р 34.11-2012 и определении hmac функции в rfc 2104 http://tools.ietf.org/html/rfc2104

Эти функции соответствуют функциям, описанным в документе **ТК26АЛГ** раздел 5.1.

```
#define CKM_GOSTR3411_2012_256_HMAC (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x014)
#define CKM_GOSTR3411_2012_512_HMAC (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x015)
```

CKM_GOSTR3411_2012_256_HMAC со значениями B = 64, L = 32 с длиной значения 32 байта

и CKM_GOSTR3411_2012_512_HMAC со значениями B = 64, L = 64 с длиной значения 64 байта соответственно. Эти механизмы не используют параметры.

Используется в C_SignInit

9. Механизм создания ключей

Для создания ключей длины 512 бит для стандарта ГОСТ Р 34.10-2012 вводится механизм

```
CKM_GOSTR3410_512_KEY_PAIR_GEN
#define CKM_GOSTR3410_512_KEY_PAIR_GEN (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x005)
```

Используется для `C_GenerateKeyPair.`

Механизм не использует параметров. Набор эллиптических параметров задается в шаблонах ключей в виде ASN закодированного OID, как описано в разделе «Наборы параметров». Если параметры не указаны, используется набор по умолчанию.

10. Механизм создания открытого ключа

Механизм создания открытого ключа из закрытого.

```
#define CKM_GOSTR3410_PUBLIC_KEY_DERIVE (CK_VENDOR_PKCS11_RU_TEAM_TC26
|0x00A)
```

Используется в `C_DeriveKey`

Используются параметры, определенные в закрытом ключе, явно не задаются.

11. Механизмы для подписи / проверки.

Механизм CKM_GOSTR3410_512 используется для реализации алгоритма подписи / проверки уже вычисленного значения хэш-функции на ключе длиной 512 бит

```
#define CKM_GOSTR3410_512 (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x006)
```

Совместный алгоритм хэширования CKM_GOSTR3411_2012_256 и подписи на ключе длиной 256 бит.

```
#define CKM_GOSTR3410_WITH_GOSTR3411_2012_256
(CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x008)
```

Совместный алгоритм хэширования CKM_GOSTR3411_2012_512 и подписи на ключе длиной 512 бит.

```
#define CKM_GOSTR3410_WITH_GOSTR3411_2012_512
(CK_VENDOR_PKCS11_RU_TEAM_TCK26 |0x009)
```

Используется в `C_SignInit` и `C_VerifyInit`, параметры используются из ключа и явно в алгоритме не задаются.

12. Механизм диверсификации ключа.

Для диверсификации ключа шифрования CKK_GOST28147 вводится механизм CKM_KDF_4357. Алгоритм диверсификации определен в **RFC 4357** в разделе 6.5.

```
#define CKM_KDF_4357                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x025)
```

Используется в `C_DeriveKey`.

Механизм использует единственный параметр UKM.

13. Механизм диверсификации ключа.

Для диверсификации ключей вводится механизм CKM_KDF_GOSTR3411_2012_256. Алгоритм определен в **ТК26АЛГ** в разделе 5.4.

```
#define CKM_KDF_GOSTR3411_2012_256          (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x026)
```

Алгоритм использует фиксированное значение в поле label –последовательность четырех байт: `0x26|0xBD|0xB8|0x78`. Механизм использует 1 параметр, который в описании алгоритма обозначен как seed.

Используется в C_DeriveKey и может применяться к ключам шифрования CKK_GOST28147 и CKK_KUZNECHIK.

14. Механизм согласования ключей

Для реализации алгоритма Диффи-Хеллмана для ключей ГОСТ Р 34.10-2012 вводится механизм выработки производного ключа CKM_GOSTR3410_2012_DERIVE.

```
#define CKM_GOSTR3410_2012_DERIVE (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x007)
```

Является алгоритмом согласования ключей VKO GOST R 34.10-2012, на основе ГОСТ Р 34.11-2012 и используется для согласования ключей ГОСТ Р34.10-2012 (256 и 512 бит) и предназначен для получения ключевого материала длины 256 бит, использующегося в криптографических протоколах.
Механизм соответствует алгоритму, описанному в документе **ТК26АЛГ** раздел 5.3.

К полученному таким образом ключу может быть применен (опционально) алгоритм диверсификации.

Параметры механизма CKM_GOSTR3410_2012_DERIVE задаются байтовым массивом, который имеет следующую структуру:

- 4 байта (little-endian, т.е. младшие байты сначала) представляют собой значение KDF. Значение определяет механизм диверсификации, один из следующих:
  - CKD_NULL - нет диверсификации
  - CKM_KDF_4357
  - CKD_CPDIVERSIFY_KDF или CKM_KDF_GOSTR3411_2012_256

- 4 байта (little-endian) задают длину открытого ключа в байтах. (64 либо 128)

- открытый ключ, длина которого определена предыдущим полем (64 (либо 128)-байтовый вектор координаты точки X и Y – два вектора длиной по 32(либо 64) байта в little-endian)

- 4 байта (little-endian) задают длину UKM (от 8 байт)

- UKM (n-байтовый вектор в little-endian) , длина определена выше.

(Константа CKD_NULL равна 1, а CKD_CPDIVERSIFY_KDF равна 9 в PKCS#11 v2.30 draft)

Используется в C_DeriveKey

15. Использование алгоритмов хэширования ГОСТ Р 34.11-2012 в механизме генерации ключей PKCS #5 PBKDF2.

Генерация ключей по алгоритму PKCS #5 PBKDF2 осуществляется в соответствии со стандартом PKCS#11(пункт 6.22 в версии 2.30). Функции C_GenerateKey и C_GenerateKeyPair вызываются с механизмом CKM_PKCS5_PBKD2. В качестве параметра механизма используется структура CK_PKCS5_PBKD2_PARAMS.

При этом для использования алгоритмов хэширования ГОСТ Р 34.11-2012 в поле

```
CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE prf;
```

структуры CK_PKCS5_PBKD2_PARAMS должно указываться одно из значений

CKP_PKCS5_PBKD2_HMAC_GOSTR3411_2012_256

CKP_PKCS5_PBKD2_HMAC_GOSTR3411_2012_512

для выбора функции хэширования 256 и 512 бит соответственно. Эти значения определены следующим образом

```
#define CKP_PKCS5_PBKD2_HMAC_GOSTR3411_2012_256
(CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x002)
#define CKP_PKCS5_PBKD2_HMAC_GOSTR3411_2012_512
(CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x003)
```

Дополнительные параметры не используются, и значение *pPrfData* должно быть *NULL*, и *ulPrfDataLen* тоже должно быть 0.

16. Псевдослучайные функции (PRF) протокола TLS.

При реализации протокола TLS с российскими алгоритмами в соответствии с документом [ТК26TLS] «ИСПОЛЬЗОВАНИЕ НАБОРОВ АЛГОРИТМОВ ШИФРОВАНИЯ НА ОСНОВЕ ГОСТ 28147-89 ДЛЯ ПРОТОКОЛА БЕЗОПАСНОСТИ ТРАНСПОРТНОГО УРОВНЯ (TLS)» используются алгоритмы PRF_TLS_GOSTR3411_2012_256 и PRF_TLS_GOSTR3411_2012_512, основанные на стандарте хэширования ГОСТ Р 34.11-2012 и описанные в документе [ТК26АЛГ] «ИСПОЛЬЗОВАНИЕ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ, СОПУТСТВУЮЩИХ ПРИМЕНЕНИЮ СТАНДАРТОВ ГОСТ Р 34.10-2012 И ГОСТ Р 34.11-2012»

Для получения значений псевдослучайных функций (PRF) в соответствии со стандартом PKCS#11(пункт 6.25 в версии 2.30) используется функция C_Derive. Для использования функций PRF_TLS_GOSTR3411_2012_256 и PRF_TLS_GOSTR3411_2012_512 введены механизмы CKM_TLS_GOST_PRF_2012_256 и CKM_TLS_GOST_PRF_2012_512 соответственно, которые определенны

```
#define CKM_TLS_GOST_PRF_2012_256 (CK_VENDOR_PKCS11_RU_TEAM_TC26
|0x016)
#define CKM_TLS_GOST_PRF_2012_512 (CK_VENDOR_PKCS11_RU_TEAM_TC26
|0x017)
```

В качестве параметров механизма используется структура CK_TLS_PRF_PARAMS. Функция C_Derive вырабатывает псевдослучайный набор байтов указанной в поле *pulOutputLen* длины, и возвращает результат не через дескриптор ключа, а через поле *pOutput.* Аргумент *phKey* не используется и должен быть NULL_PTR.

17. Создание ключа master_secret.

В реализации протокола TLS в соответствии с документом [ТК26TLS] «ИСПОЛЬЗОВАНИЕ НАБОРОВ АЛГОРИТМОВ ШИФРОВАНИЯ НА ОСНОВЕ ГОСТ 28147-89 ДЛЯ ПРОТОКОЛА БЕЗОПАСНОСТИ ТРАНСПОРТНОГО УРОВНЯ (TLS)» используется ключ master_secret (48 байт).

Этот ключ никогда не используется в чистом виде, используется только результат применения функции хэширования к этому ключу. Поэтому из соображений оптимизации и увеличения безопасности вводится объединенный механизм

```
#define CKM_TLS_GOST_MASTER_KEY_DERIVE_2012_256
(CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x018)
```

В функцию C_DeriveKey передается механизм CKM_TLS_GOST_MASTER_KEY_DERIVE_2012_256 и ключ premaster_secret. В качестве параметра механизма выступает структура CK_SSL3_RANDOM_DATA, позволяющая передать случайные данные сервера и клиента.

Реализация вырабатывает ключ master_secret и вычисляет значение хэш-функции по алгоритму ГОСТ Р 34.11-2012, которое становится значением ключа CKA_VALUE. Механизм устанавливает в созданный ключ длину ключа CKA_VALUE_LEN 32 байта, в качестве атрибута CKA_CLASS значение CKO_SECRET_KEY, CKA_KEY_TYPE получает значение CKK_GOST28147.

Использование атрибутов CKA_SENSITIVE, CKA_ALWAYS_SENSITIVE, CKA_EXTRACTABLE, CKA_NEVER_EXTRACTABLE описано в PKCS#11(пункт 6.25.5 в версии 2.30) и не изменяется.

Для этого механизма поля ulMinKeySize и ulMaxKeySize структуры CK_MECHANISM_INFO равны 32 байтам.

18. Использование механизмов TLS 1.2.

В стандарте PKCS#11 v 2.40 для реализации протокола TLS создан набор механизмов.

```
CKM_TLS12_MASTER_KEY_DERIVE
CKM_TLS12_MASTER_KEY_DERIVE_DH
CKM_TLS12_KEY_AND_MAC_DERIVE
CKM_TLS12_MAC
```

Эти механизмы предполагают использование структур содержащих параметры механизма

```
CK_TLS12_MASTER_KEY_DERIVE_PARAMS
CK_TLS12_KEY_MAT_PARAMS
CK_TLS_KDF_PARAMS
CK_TLS_MAC_PARAMS
```

Эти структуры содержат поле prfHashMechanism (или prfMechanism). Для использования этих механизмов при реализации протокола TLS с российскими алгоритмами следует в

качестве значения поля prfHashMechanism указывать механизм CKM_TLS_GOST_PRF_2012_256.

19. Создание ключа ГОСТ Р 34.12-2015.

Для создания ключей для стандарта ГОСТ Р 34.12-2015 с длиной блока 128 бит вводится механизм CKM_KUZNECHIK_KEY_GEN.

```
#define CKM_KUZNECHIK_KEY_GEN                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x019)
```

Используется для `C_GenerateKey`.

Механизм не требует параметров.

Создает ключ длиной 256 бит. CKA_VALUE_LEN 32 байта.

20. Алгоритм шифрование Кузнечик.

Для шифрования данных в соответствии со стандартами ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 по алгоритму Кузнечик с длиной блока 128 бит в разных режимах предназначены следующие механизмы:

```
CKM_KUZNECHIK_ECB
CKM_KUZNECHIK_CTR
CKM_KUZNECHIK_CFB
CKM_KUZNECHIK_OFB
CKM_KUZNECHIK_CBC
```

Механизмы используются в функции C_EncryptInit и C_DecryptInit для начала single- and multiple-part encryption and decryption. Они используют для шифрования ключ, у которого в качестве класса объекта указан CKO_SECRET_KEY и в атрибуте CKA_KEY_TYPE стоит значение CKK_KUZNECHIK**.**

20.1 Шифрование в режиме простой замены

Для шифрования по алгоритму Кузнечик по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме простой замены с длиной блока 128 бит служит механизм CKM_KUZNECHIK_ECB.

```
#define CKM_KUZNECHIK_ECB                        (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01A)
```

Механизм не требует параметров.

Длина шифруемой информации должна быть кратна длине блока.

Этот механизм может применяться для экспорта / импорта ключей.

20.2 Шифрование в режиме гаммирования.

Для шифрования по алгоритму Кузнечик по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме гаммирования с длиной блока 128 бит служит механизм CKM_KUZNECHIK_CTR.

```
#define CKM_KUZNECHIK_CTR                        (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01B)
```

Механизм использует единственный параметр – синхропосылку (вектор инициализации), которая имеет длину 8 байт.

Ограничения на длину шифруемой информации не накладываются.

20.3 Шифрование в режиме гаммирования с обратной связью по шифртексту.

Для шифрования по алгоритму Кузнечик по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме гаммирования с обратной связью по шифртексту с длиной блока 128 бит служит механизм CKM_KUZNECHIK_CFB.

```
#define CKM_KUZNECHIK_CFB                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01C)
```

Механизм использует единственный параметр – синхропосылку (вектор инициализации), которая по стандарту ГОСТ Р 34.13-2015 имеет длину 16 байт. Размер зацепления равен длине блока.

20.4 Шифрование в режиме гаммирования с обратной связью по выходу.

Для шифрования по алгоритму Кузнечик по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме гаммирования с обратной связью по выходу с длиной блока 128 бит служит механизм CKM_KUZNECHIK_OFB.

```
#define CKM_KUZNECHIK_OFB                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01D)
```

Механизм использует единственный параметр – синхропосылку (вектор инициализации), которая по стандарту ГОСТ Р 34.13-2015 имеет длину 16 байт. Размер зацепления равен длине блока.

20.5 Шифрование в режиме простой замены с зацеплением.

Для шифрования по алгоритму Кузнечик по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме простой замены с зацеплением с длиной блока 128 бит служит механизм CKM_KUZNECHIK_CBC.

```
#define CKM_KUZNECHIK_CBC                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01E)
```

Механизм использует единственный параметр – синхропосылку (вектор инициализации), которая по стандарту ГОСТ Р 34.13-2015 имеет длину 16 байт. Размер зацепления равен длине блока.

21. Алгоритм шифрование Магма.

Для шифрования данных в соответствии со стандартами ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 по алгоритму Магма с длиной блока 64 бит предназначены следующие механизмы:

```
CKM_MAGMA_CTR
CKM_MAGMA_CFB
CKM_MAGMA_OFB
CKM_MAGMA_CBC
```

Механизмы используются в функции C_EncryptInit и C_DecryptInit для начала single- and multiple-part encryption and decryption. Они используют для шифрования ключ, у которого в качестве класса объекта указан CKO_SECRET_KEY и в атрибуте CKA_KEY_TYPE стоит значение CKK_GOST28147.

Во всех режимах должен использоваться алгоритм мешинга, описанный в **RFC 4357** в разделе 2.3.2 и подстановка описанная в **ГОСТ Р 34.12-2015** в разделе 5.1.1.

21.1 Шифрование в режиме гаммирования.

Для шифрования по алгоритму Магма по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме гаммирования с длиной блока 64 бит служит механизм CKM_MAGMA_CTR.

```
#define CKM_MAGMA_CTR                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x020)
```

Механизм использует единственный параметр – синхропосылку (вектор инициализации), которая имеет длину 4 байт.

Ограничения на длину шифруемой информации не накладываются.

21.2 Шифрование в режиме гаммирования с обратной связью по шифртексту.

Для шифрования по алгоритму Магма по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме гаммирования с обратной связью по шифртексту с длиной блока 64 бит служит механизм CKM_MAGMA_CFB.

```
#define CKM_MAGMA_CFB                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x021)
```

Механизм использует единственный параметр – синхропосылку (вектор инициализации), которая имеет длину 8 байт. Размер зацепления равен длине блока.

21.3 Шифрование в режиме гаммирования с обратной связью по выходу.

Для шифрования по алгоритму Магма по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме гаммирования с обратной связью по выходу с длиной блока 64 бит служит механизм CKM_MAGMA_OFB.

```
#define CKM_MAGMA_OFB                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x022)
```

Механизм использует единственный параметр – синхропосылку (вектор инициализации), которая имеет длину 8 байт. Размер зацепления равен длине блока.

21.4Шифрование в режиме простой замены с зацеплением.

Для шифрования по алгоритму Магма по стандарту ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 в режиме простой замены с зацеплением с длиной блока 64 бит служит механизм CKM_MAGMA_CBC.

```
#define CKM_MAGMA_CBC                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x023)
```

Механизм использует единственный параметр – синхропосылку (вектор инициализации), которая имеет длину 8 байт. Размер зацепления равен длине блока.

22. Подпись и проверка подписи по стандарту ГОСТ Р 34.13-2015.

Для подписи и проверки подписи по стандарту ГОСТ Р 34.13-2015 в режиме выработки имитовставки

```
#define CKM_KUZNECHIK_MAC                (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01F)
#define CKM_MAGMA_MAC                    (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x024)
```

Механизмы не используют параметров. Ограничений на длину подписываемого сообщения нет. Размер получаемого значения подписи равен 8 байт.

1. Приложение 1. Заголовочный файл pkcs11gost.h.

```
#define NSSCK_VENDOR_PKCS11_RU_TEAM 0xD4321000 /* 0x80000000 | 0x54321000 */
#define CK_VENDOR_PKCS11_RU_TEAM_TC26           NSSCK_VENDOR_PKCS11_RU_TEAM

/* GOST KEY TYPES */
#define CKK_GOSTR3410_256                       CKK_GOSTR3410
#define CKK_GOSTR3410_512                       (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x003)
#define CKK_KUZNECHIK                           (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x004)

/* GOST OBJECT ATTRIBUTES */
#define CKA_GOSTR3410_256PARAMS                 CKA_GOSTR3410PARAMS
#define CKA_GOSTR3410_512PARAMS                 (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x004)

/* PKCS #5 PRF Functions */
#define CKP_PKCS5_PBKD2_HMAC_GOSTR3411_2012_256 (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x002)
#define CKP_PKCS5_PBKD2_HMAC_GOSTR3411_2012_512 (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x003)

/* GOST MECHANISMS */
#define CKM_GOSTR3410_256_KEY_PAIR_GEN          CKM_GOSTR3410_KEY_PAIR_GEN
#define CKM_GOSTR3410_512_KEY_PAIR_GEN          (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x005)
#define CKM_GOSTR3410_256                       CKM_GOSTR3410
#define CKM_GOSTR3410_512                       (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x006)
#define CKM_GOSTR3410_2012_DERIVE               (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x007)
#define CKM_GOSTR3410_12_DERIVE                 CKM_GOSTR3410_2012_DERIVE
#define CKM_GOSTR3410_WITH_GOSTR3411_94         CKM_GOSTR3410_WITH_GOSTR3411
#define CKM_GOSTR3410_WITH_GOSTR3411_2012_256   (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x008)
#define CKM_GOSTR3410_WITH_GOSTR3411_2012_512   (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x009)
#define CKM_GOSTR3410_WITH_GOSTR3411_12_256     CKM_GOSTR3410_WITH_GOSTR3411_2012_256
#define CKM_GOSTR3410_WITH_GOSTR3411_12_512     CKM_GOSTR3410_WITH_GOSTR3411_2012_512
#define CKM_GOSTR3410_PUBLIC_KEY_DERIVE         (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x00A)

#define CKM_GOSTR3411_94                        CKM_GOSTR3411
#define CKM_GOSTR3411_2012_256                  (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x012)
#define CKM_GOSTR3411_2012_512                  (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x013)
#define CKM_GOSTR3411_12_256                    CKM_GOSTR3411_2012_256
#define CKM_GOSTR3411_12_512                    CKM_GOSTR3411_2012_512
#define CKM_GOSTR3411_94_HMAC                   CKM_GOSTR3411_HMAC
#define CKM_GOSTR3411_2012_256_HMAC             (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x014)
#define CKM_GOSTR3411_2012_512_HMAC             (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x015)
#define CKM_GOSTR3411_12_256_HMAC               CKM_GOSTR3411_2012_256_HMAC
#define CKM_GOSTR3411_12_512_HMAC               CKM_GOSTR3411_2012_512_HMAC
#define CKM_TLS_GOST_PRF_2012_256               (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x016)
#define CKM_TLS_GOST_PRF_2012_512               (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x017)
#define CKM_TLS_GOST_PRE_MASTER_KEY_GEN         CKM_GOST28147_KEY_GEN
#define CKM_TLS_GOST_MASTER_KEY_DERIVE_2012_256 (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x018)

#define CKM_KUZNECHIK_KEY_GEN                   (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x019)
#define CKM_KUZNECHIK_ECB                       (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01A)
#define CKM_KUZNECHIK_CTR                       (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01B)
#define CKM_KUZNECHIK_CFB                       (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01C)
#define CKM_KUZNECHIK_OFB                       (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01D)
#define CKM_KUZNECHIK_CBC                       (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01E)
#define CKM_KUZNECHIK_MAC                       (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x01F)

#define CKM_MAGMA_ECB                           CKM_GOST28147_ECB
#define CKM_MAGMA_CTR                           (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x020)
#define CKM_MAGMA_CFB                           (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x021)
#define CKM_MAGMA_OFB                           (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x022)
#define CKM_MAGMA_CBC                           (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x023)
#define CKM_MAGMA_MAC                           (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x024)

#define CKM_KDF_4357                            (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x025)
#define CKM_KDF_GOSTR3411_2012_256              (CK_VENDOR_PKCS11_RU_TEAM_TC26 |0x026)
```