Расширение PKCS#11 для использования российских криптографических алгоритмов

Листов 41

Москва, 2008 г.

Аннотация

Данный документ определяет расширение спецификаций PKCS#11 для использования ключей и криптографических алгоритмов ГОСТ 28147-89, ГОСТ Р 34.11-94, ГОСТ Р 34.10-2001, а также алгоритмов, построенных на их основе, в соответствии с международными рекомендациями RFC 4357: «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».

Оглавление

1 Дополнительные определения для PKCS#11

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*/
```

1.1 Типы ключей

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

```
#define CKK_GOSTR3410   (NSSCK_VENDOR_PKSC11_RU_TEAM | 0x000)
#define CKK_GOSTR3411   (NSSCK_VENDOR_PKSC11_RU_TEAM | 0x001)
#define CKK_GOST28147   (NSSCK_VENDOR_PKSC11_RU_TEAM | 0x002)
```

1.2 Типы атрибутов

Данное расширение PKCS#11 вводит следующие определения в пространстве значений типов атрибутов:

```
#define CKA_GOSTR3410PARAMS     (NSSCK_VENDOR_PKSC11_RU_TEAM |0x001)
#define CKA_GOSTR3411PARAMS     (NSSCK_VENDOR_PKSC11_RU_TEAM |0x002)
#define CKA_GOST28147PARAMS     (NSSCK_VENDOR_PKSC11_RU_TEAM |0x003)
```

Формат значений атрибутов указанных типов – DER-представление объектных идентификаторов, указывающих на параметры алгоритмов, в соответствии с идентификаторами алгоритмов прикладного уровня.

1.3 Механизмы

Дополнительные определения в пространстве значений идентификаторов механизмов приведены в Приложении 1. **Таблица 1** описывает возможности применения отдельных механизмов для Cryptoki-вызовов.

Таблица 1. Функции и механизмы.

Механизм	Функция						
	Encrypt & Decrypt	Sign & Verify	SR & VR	Digest	Gen Key/ Key Pair	Wrap & Unwrap	Derive
CKM_GOST28147_KEY_GEN					+		
CKM_GOST28147_ECB	+					+	
CKM_GOST28147	+					+	
CKM_GOST28147_MAC		+					
CKM_GOST28147_KEY_WRAP						+	
CKM_GOSTR3411				+			
CKM_GOSTR3411_HMAC		+					
CKM_GOSTR3410_KEY_PAIR_GEN					+		
CKM_GOSTR3410		+[1]					
CKM_GOSTR3410_WITH_GOSTR3411		+					
CKM_GOSTR3410_KEY_WRAP						+	
CKM_GOSTR3410_DERIVE							+
CKM_TLS_GOST_PRF							+
CKM_TLS_PRE_MASTER_KEY_GEN					+		
CKM_TLS_GOST_MASTER_KEY_DERIVE							+
CKM_TLS_GOST_KEY_AND_MAC_DERIVE							+

[1] Поддерживается только одношаговая операция, что соответствует режиму обработки single-part в терминологии PKCS#11

2 Объекты

2.1 ГОСТ 28147-89

2.1.1 Ключ

Для представления ключей шифрования и выработки имитовставки ГОСТ 28147-89 используются объекты класса **CKO_SECRET_KEY** с типом ключа **CKK_GOST28147**.

Атрибуты, соответствующие объекту в дополнение к атрибутам, определенным для объектов класса **CKO_SECRET_KEY**:

Атрибут	Тип данных	Значение
CKA_VALUE[1,4,6,7]	Массив байт	Значение ключа – 32-байтовый вектор в little-endian представлении
CKA_GOST28147PARAMS[1,3,5]	Массив байт	DER-закодированное значение идентификатора используемых параметров ГОСТ 28147-89. При использовании ключа необходимо наличие объекта параметров (класс **CKO_DOMAIN_PARAMETERS**, тип ключа **CKK_GOST28147**) с идентичным значением атрибута **CKA_OBJECT_ID**

[1, 3, 4, 5, 6, 7] см. Таблицу 15 **[1]**

Пример шаблона для создания ключа с тестовыми параметрами:

```
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_GOST28147PARAMS, params_oid, sizeof(params_oid)},
    {CKA_VALUE, value, sizeof(value)}
};
```

2.1.2 Параметры

Для представления криптографических параметров шифрования и выработки имитовставки ГОСТ 28147-89 используются объекты класса **CKO_DOMAIN_PARAMETERS** с типом ключа **CKK_GOST28147**.

Атрибуты, соответствующие объекту в дополнение к атрибутам, определенным для объектов класса **CKO_DOMAIN_PARAMETERS**:

Атрибут	Тип данных	Значение
CKA_VALUE[1]	Массив байт	DER-закодированное значение параметров как представлено в RFC 4357 **[5]** раздел 8.1 типом *Gost28147-89-ParamSetParameters*
CKA_OBJECT_ID[1]	Массив байт	DER-закодированное значение идентификатора параметров шифрования и выработки имитовставки ГОСТ 28147-89

[1] см. Таблицу 15 **[1]**

Следует отметить, что некоторые реализации токена могут не поддерживать установку и/или извлечение криптографических параметров. Программы, работающие с объектами криптографических параметров, должны быть готовы к недоступности значения атрибута **CKA_VALUE**.

Пример шаблона для создания тестовых криптографических параметров ГОСТ 28147-89:

```
CK_OBJECT_CLASS class = CKO_DOMAIN_PARAMETERS;
CK_KEY_TYPE keyType = CKK_GOST28147;
CK_UTF8CHAR label[] = “A GOST 28147-89 cryptographic parameters object”;
```

```
CK_BYTE oid[] = {0x06, 0x07, 0x2a, 0x85, 0x03, 0x02, 0x02, 0x1f, 0x00};
CK_BYTE value[] = {
  0x30,0x62,
  0x04,0x40, /* eUZ */
  0x4c,0xde,0x38,0x9c,0x29,0x89,0xef,0xb6,0xff,0xeb,0x56,0xc5,0x5e,0xc2,0x9b,0x02,
  0x98,0x75,0x61,0x3b,0x11,0x3f,0x89,0x60,0x03,0x97,0x0c,0x79,0x8a,0xa1,0xd5,0x5d,
  0xe2,0x10,0xad,0x43,0x37,0x5d,0xb3,0x8e,0xb4,0x2c,0x77,0xe7,0xcd,0x46,0xca,0xfa,
  0xd6,0x6a,0x20,0x1f,0x70,0xf4,0x1e,0xa4,0xab,0x03,0xf2,0x21,0x65,0xb8,0x44,0xd8,
  0x02,0x01,0x00, /* mode */
  0x02,0x01,0x40, /* shiftBits */
  0x30,0x0b,0x06,0x07,0x2a,0x85,0x03,0x02,0x02,0x0e,0x00,0x05,0x00 /*keyMeshing*/
};
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_OBJECT_ID, oid, sizeof(oid)},
    {CKA_VALUE, value, sizeof(value)}
};
```

2.2 ГОСТ Р 34.11-94

2.2.1 Параметры

Для представления криптографических параметров хэш-функции ГОСТ Р 34.11-94 используются объекты класса **CKO_DOMAIN_PARAMETERS** с типом ключа **CKK_GOSTR3411**.

Атрибуты, соответствующие объекту, в дополнение к атрибутам, определенным для объектов класса **CKO_DOMAIN_PARAMETERS**:

Атрибут	Тип данных	Значение
CKA_VALUE[1]	Массив байт	DER-закодированное значение параметров как представлено в RFC 4357 **[5]** раздел 8.2 типом *GostR3411-94-ParamSetParameters*
CKA_OBJECT_ID[1]	Массив байт	DER-закодированное значение идентификатора параметров хэш-функции ГОСТ Р34.11-94

[1] см. таблицу 15 **[1]**

Следует отметить, что некоторые реализации токена могут не поддерживать установку и/или извлечение криптографических параметров. Программы, работающие с объектами криптографических параметров, должны быть готовы к недоступности значения атрибута **CKA_VALUE**.

Пример шаблона для создания тестовых криптографических параметров ГОСТ Р 34.11-94:

```
CK_OBJECT_CLASS class = CKO_DOMAIN_PARAMETERS;
CK_KEY_TYPE keyType = CKK_GOSTR3411;
CK_UTF8CHAR label[] = “A GOST R34.11-94 cryptographic parameters object”;
CK_BYTE oid[] = {0x06, 0x07, 0x2a, 0x85, 0x03, 0x02, 0x02, 0x1e, 0x00};
CK_BYTE value[] = {
0x30,0x64,
0x04,0x40, /* hUZ */
```

```
0x4e,0x57,0x64,0xd1,0xab,0x8d,0xcb,0xbf,0x94,0x1a,0x7a,0x4d,0x2c,0xd1,0x10,0x10,
0xd6,0xa0,0x57,0x35,0x8d,0x38,0xf2,0xf7,0x0f,0x49,0xd1,0x5a,0xea,0x2f,0x8d,0x94,
0x62,0xee,0x43,0x09,0xb3,0xf4,0xa6,0xa2,0x18,0xc6,0x98,0xe3,0xc1,0x7c,0xe5,0x7e,
0x70,0x6b,0x09,0x66,0xf7,0x02,0x3c,0x8b,0x55,0x95,0xbf,0x28,0x39,0xb3,0x2e,0xcc,
0x04,0x20, /* h0 */
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};
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_OBJECT_ID, oid, sizeof(oid)},
    {CKA_VALUE, value, sizeof(value)}
};
```

2.3 ГОСТ Р 34.10-2001

2.3.1 Ключ проверки

Для представления ключей проверки подписи ГОСТ Р 34.10-2001 используются объекты класса **CKO_PUBLIC_KEY** с типом ключа **CKK_GOSTR3410**.

Атрибуты, соответствующие объекту, в дополнение к атрибутам, определенным для объектов класса **CKO_PUBLIC_KEY**:

Атрибут	Тип данных	Значение
CKA_VALUE[1,4]	Массив байт	64-байтовый вектор – значение ключа, координаты точки X и Y – два вектора длиной по 32 байта в little-endian байтовом порядке, младшие байты сначала

CKA_GOSTR3410PARAMS[1,3]	Массив байт	DER-закодированное значение идентификатора параметров ЭЦП ГОСТ Р 34.10-2001. При использовании ключа необходимо наличие объекта параметров (класс **CKO_DOMAIN_PARAMETERS**, тип ключа **CKK_GOSTR3410**) с идентичным значением атрибута **CKA_OBJECT_ID**
CKA_GOSTR3411PARAMS[1,3,8]	Массив байт	DER-закодированное значение идентификатора используемых по умолчанию параметров хэш-функции ГОСТ Р 34.11-94. Применение ключа в механизмах с операцией хэширования при использовании параметров «по умолчанию» предполагает наличие объекта параметров (класс **CKO_DOMAIN_PARAMETERS**, тип ключа **CKK_GOSTR3411**) с идентичным значением атрибута **CKA_OBJECT_ID**
CKA_GOST28147PARAMS[8]	Массив байт	DER-закодированное значение идентификатора используемых по умолчанию параметров шифрования ГОСТ 28147-89. При использовании ключа в механизмах использующих операцию шифрования с параметрами «по умолчанию» необходимо наличие объекта параметров (класс **CKO_DOMAIN_PARAMETERS**, тип ключа **CKK_GOST28147**) с идентичным значением атрибута **CKA_OBJECT_ID**. Значение этого атрибута может быть пустым

[1, 3, 4, 8] см. таблицу 15 **[1]**

Пример шаблона для создания ключа с тестовыми параметрами:

```
CK_OBJECT_CLASS class = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_GOSTR3410;
CK_UTF8CHAR label[] = "A GOST R34.10-2001 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_GOSTR3410PARAMS, gostR3410params_oid, sizeof(gostR3410params_oid)},
    {CKA_GOSTR3411PARAMS, gostR3411params_oid, sizeof(gostR3411params_oid)},
    {CKA_GOST28147PARAMS, gost28147params_oid, sizeof(gost28147params_oid)},
    {CKA_VALUE, value, sizeof(value)}
};
```

2.3.2 Ключ подписи

Для представления ключей подписи ГОСТ Р 34.10-2001 используются объекты класса **CKO_PRIVATE_KEY** с типом ключа **CKK_GOSTR3410**.

Атрибуты, соответствующие объекту, в дополнение к атрибутам, определенным для объектов класса **CKO_PRIVATE_KEY**:

Атрибут	**Тип данных**	**Значение**
CKA_VALUE[1,4,6,7]	Массив байт	32-байтовый вектор – значение ключа в little-endian байтовом порядке, младшие байты сначала
CKA_GOSTR3410PARAMS[1,4,6]	Массив байт	DER-закодированное значение идентификатора параметров ЭЦП ГОСТ Р 34.10-2001. При использовании ключа необходимо наличие объекта параметров (класс **CKO_DOMAIN_PARAMETERS**, тип ключа **CKK_GOSTR3410**) с идентичным значением атрибута **CKA_OBJECT_ID**

CKA_GOSTR3411PARAMS[1,4,6,8]	Массив байт	DER-закодированное значение идентификатора используемых по умолчанию параметров хэш-функции ГОСТ Р 34.11-94. При использовании ключа в механизмах использующих операцию хэширования с параметрами "по умолчанию" необходимо наличие объекта параметров (класс **CKO_DOMAIN_PARAMETERS**, тип ключа **CKK_GOSTR3411**) с идентичным значением атрибута **CKA_OBJECT_ID**
CKA_GOST28147PARAMS[4,6,8]	Массив байт	DER-закодированное значение идентификатора используемых по умолчанию параметров шифрования ГОСТ 28147-89. При использовании ключа в механизмах использующих операцию шифрования с параметрами "по умолчанию" необходимо наличие объекта параметров (класс **CKO_DOMAIN_PARAMETERS**, тип ключа **CKK_GOST28147**) с идентичным значением атрибута **CKA_OBJECT_ID**. Значение этого атрибута может быть пустым

[1, 4, 6, 7, 8] см. таблицу 15 **[1]**

Следует отметить, что при генерации ключа подписи атрибуты криптографических параметров не указываются в шаблоне ключа подписи. Это следует из того, что ключ подписи генерируется только в составе ключевой пары, и атрибуты криптографических параметров для ключевой пары указываются в шаблоне ключа проверки.

Пример шаблона для создания ключа с тестовыми параметрами:

```
CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_GOSTR3410;
CK_UTF8CHAR label[] = “A GOST R34.10-2001 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_GOSTR3410PARAMS, gostR3410params_oid, sizeof(gostR3410params_oid)},
    {CKA_GOSTR3411PARAMS, gostR3411params_oid, sizeof(gostR3411params_oid)},
    {CKA_GOST28147PARAMS, gost28147params_oid, sizeof(gost28147params_oid)},
    {CKA_VALUE, value, sizeof(value)}
};
```

2.3.3 Параметры

Для представления криптографических параметров ГОСТ Р 34.10-2001 используются объекты класса **CKO_DOMAIN_PARAMETERS** с типом ключа **CKK_GOSTR3410**.

Атрибуты, соответствующие объекту, в дополнение к атрибутам, определенным для объектов класса **CKO_DOMAIN_PARAMETERS**:

Атрибут	**Тип данных**	**Значение**
CKA_VALUE[1]	Массив байт	DER-закодированное значение параметров как представлено в RFC 4357 **[5]** раздел 8.4 типом *GostR3410-2001-ParamSetParameters*

CKA_OBJECT_ID[1]	Массив байт	DER-закодированное значение идентификатора параметров ЭЦП ГОСТ Р 34.10-2001

[1] см. таблицу 15 **[1]**

Следует отметить, что некоторые реализации токена могут не поддерживать установку и/или извлечение криптографических параметров. Программы, работающие с объектами криптографических параметров, должны быть готовы к недоступности значения атрибута **CKA_VALUE**.

Пример шаблона для создания тестовых криптографических параметров ГОСТ Р 34.10-2001:

```
CK_OBJECT_CLASS class = CKO_DOMAIN_PARAMETERS;
CK_KEY_TYPE keyType = CKK_GOSTR3410;
CK_UTF8CHAR label[] = "A GOST R34.10-2001 cryptographic parameters object";
CK_BYTE oid[] = {0x06, 0x07, 0x2a, 0x85, 0x03, 0x02, 0x02, 0x23, 0x00};
CK_BYTE value[] = {
0x30,0x81,0x90,
0x02,0x01,0x07,
0x02,0x20,
0x5f,0xbf,0xf4,0x98,0xaa,0x93,0x8c,0xe7,0x39,0xb8,0xe0,0x22,0xfb,0xaf,0xef,0x40,
0x56,0x3f,0x6e,0x6a,0x34,0x72,0xfc,0x2a,0x51,0x4c,0x0c,0xe9,0xda,0xe2,0x3b,0x7e,
0x02,0x21,0x00,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x04,0x31,
0x02,0x21,0x00,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x50,0xfe,0x8a,0x18,0x92,0x97,0x61,0x54,0xc5,0x9c,0xfc,0x19,0x3a,0xcc,0xf5,0xb3,
0x02,0x01,0x02,
0x02,0x20,
0x08,0xe2,0xa8,0xa0,0xe6,0x51,0x47,0xd4,0xbd,0x63,0x16,0x03,0x0e,0x16,0xd1,0x9c,
0x85,0xc9,0x7f,0x0a,0x9c,0xa2,0x67,0x12,0x2b,0x96,0xab,0xbc,0xea,0x7e,0x8f,0xc8
};
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_OBJECT_ID, oid, sizeof(oid)},
    {CKA_VALUE, value, sizeof(value)}
};
```

3 Механизмы

3.1 ГОСТ 28147-89

3.1.1 Генерация ключа шифрования и выработки имитовставки

Генерация ключа шифрования и выработки имитовставки ГОСТ 28147-89 осуществляется с помощью механизма **CKM_GOST28147_KEY_GEN**.

Этот механизм не имеет параметра.

Механизм устанавливает атрибуты **CKA_CLASS**, **CKA_KEY_TYPE** и **CKA_VALUE** для создаваемого объекта ключа. Шаблон может содержать атрибуты, разрешенные для объектов класса **CKO_SECRET_KEY**.

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.1.2 Шифрование в режиме простой замены

Шифрование ГОСТ 28147-89 в режиме простой замены осуществляется с помощью механизма **CKM_GOST28147_ECB**. Длина входных данных для каждой операции шифрования должна быть кратной длине блока, т.е. 8-ми байтам. Блок подстановки, используемый при шифровании, определяется идентификатором, содержащимся в атрибуте **CKA_GOST28147PARAMS** используемого ключа.

Этот механизм не имеет параметра.

Механизм может быть использован для шифрования секретных ключей любых типов. При операции **C_WrapKey**, значение атрибута ключа **CKA_VALUE** дополняется нулевыми байтами так, чтобы его длина стала кратной 8-ми байтам и

потом зашифровывается. При операции **C_UnwrapKey**, результат расшифрования приводится к длине соответствующей значению атрибута **CKA_KEY_TYPE** или **CKA_VALUE_LEN** шаблона создаваемого ключа и устанавливается как значение атрибута **CKA_VALUE** создаваемого ключа.

Длины ключей и данных для механизма **CKM_GOST28147_ECB**:

Функция	**Тип ключа**	**Длина входных данных**	**Длина выходных данных**
C_Encrypt	**CKK_GOST28147**	Кратна восьми	Равна длине входных данных
C_Decrypt	**CKK_GOST28147**	Кратна восьми	Равна длине входных данных
C_WrapKey	**CKK_GOST28147**	Произвольная	Равна длине входных данных, увеличенной до значения кратного восьми
C_UnwrapKey	**CKK_GOST28147**	Кратна восьми	Зависит от типа ключа, подлежащего расшифрованию

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.1.3 Шифрование

Шифрование ГОСТ 28147-89 во всех режимах, кроме простой замены, осуществляется с помощью механизма **CKM_GOST28147**. Шифрование производится в соответствии с криптографическими параметрами (режим шифрования, блок подстановки, алгоритм смены ключа и т.д.), определяемыми идентификатором, содержащимся в атрибуте **CKA_GOST28147PARAMS** используемого ключа. Описание различных режимов шифрования и алгоритмов смены ключа находится в ГОСТ 28147-89 **[2]** и RFC 4357 **[5]** раздел 2.

Параметром этого механизма является 8-байтовый инициализационный вектор. Параметр механизма также может отсутствовать, в этом случае используется

нулевой инициализационный вектор.

Механизм может быть использован для шифрования секретных ключей любых типов. При операции **C_WrapKey**, результатом является зашифрованное значение атрибута ключа **CKA_VALUE**. При операции **C_UnwrapKey**, результат расшифрования устанавливается как значение атрибута **CKA_VALUE** создаваемого ключа.

Длины ключей и данных для механизма **CKM_GOST28147**:

Функция	**Тип ключа**	**Длина входных данных**	**Длина выходных данных**
C_Encrypt	**CKK_GOST28147**	Произвольная	Для режимов гаммирования и гаммирования с обратной связью – равна длине входных данных, для режима CBC – равна длине входных данных, увеличенной до значения кратного восьми
C_Decrypt	**CKK_GOST28147**	Произвольная	
C_WrapKey	**CKK_GOST28147**	Произвольная	
C_UnwrapKey	**CKK_GOST28147**	Произвольная	

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.1.4 Выработка и проверка имитовставки

Выработка и проверка имитовставки ГОСТ 28147-89 осуществляется с помощью механизма **CKM_GOST28147_MAC**.

Выработка имитовставки производится в соответствии с криптографическими параметрами, определяемыми идентификатором, содержащимся в атрибуте **CKA_GOST28147PARAMS** используемого ключа. При этом используются блок подстановки и алгоритм смены ключа, указанные в криптографических параметрах. Описание различных алгоритмов смены ключа приведено в RFC 4357 **[5]** раздел 2.3. Значение имитовставки – первые четыре байта результата.

Параметром этого механизма является 8-байтовый инициализационный вектор. Параметр механизма также может отсутствовать, в этом случае используется нулевой инициализационный вектор.

Длины ключей и данных для механизма **CKM_GOST28147_MAC**:

Функция	Тип ключа	Длина данных	Длина имитовставки
C_Sign	**CKK_GOST28147**	Произвольная	Четыре байта
C_Verify	**CKK_GOST28147**	Произвольная	Четыре байта

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.1.5 Шифрование ключей ГОСТ 28147-89 при помощи ключей ГОСТ 28147-89

Шифрование ключей ГОСТ 28147-89 с использованием ключей, выработанных в соответствии с ГОСТ 28147-89, осуществляется при помощи механизма **CKM_GOST28147_KEY_WRAP**.

При операции **C_WrapKey** производится выработка имитовставки от значения атрибута **CKA_VALUE** зашифровываемого ключа и зашифрование этого же значения атрибута **CKA_VALUE** в режиме простой замены. Результатом является последовательная запись 32-байтового зашифрованного ключа и четырех байт имитовставки.

При операции **C_UnwrapKey** производится расшифрование начальных 32 байт входных данных в режиме простой замены и выработка имитовставки от полученного значения. Четыре байта имитовставки сравниваются с последними четырьмя байтами входных данных, в случае несовпадения ключ отвергается. Результат расшифрования устанавливается как значение атрибута **CKA_VALUE**

создаваемого ключа.

Параметром этого механизма является 8-байтовый инициализационный вектор выработки имитовставки. Параметр механизма также может отсутствовать, в этом случае используется нулевой инициализационный вектор.

Длины ключей и данных для механизма **CKM_GOST28147_KEY_WRAP**:

Функция	**Тип ключа**	**Длина входных данных**	**Длина выходных данных**
C_WrapKey	**CKK_GOST28147**	32 байта	36 байтов
C_UnwrapKey	**CKK_GOST28147**	36 байтов	32 байта

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.2 ГОСТ Р 34.11-94

3.2.1 Вычисление хэш-функции

Вычисление хэш-функции ГОСТ Р 34.11-94 осуществляется с помощью механизма **CKM_GOSTR3411**.

Параметром этого механизма является DER-закодированный идентификатор криптографических параметров ГОСТ Р 34.11-94, используемых при вычислении хэш-функции. Параметр механизма может отсутствовать, при этом будут использованы криптографические параметры, определяемые идентификатором *id-GostR3411-94-CryptoProParamSet* (см. RFC 4357 **[5]** раздел 11.2).

Результатом является 32-байтовый вектор, содержащий значение хэш-функции. При одношаговой операции хэширования, входные данные и результат могут находиться в одной области памяти.

Длины данных для механизма **CKM_GOSTR3411**:

Функция	Длина данных	Длина хэш-вектора
C_Digest	Произвольная	32 байта

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.2.2 Вычисление значения хэш-функции на ключе (HMAC) при помощи ГОСТ Р 34.11-94

Вычисление хэш-функции на ключе (HMAC) с использованием хэш-функции ГОСТ Р 34.11-94 осуществляется с помощью механизма **CKM_GOSTR3411_HMAC**. Алгоритм представлен в RFC 2104 **[9]** и использует хэш-функцию ГОСТ Р 34.11-94 с длиной HMAC-блока равной 32 (B=32, L=32 – см. RFC 2104 **[9]** раздел 2, RFC 4357 **[5]** раздел 3). Ключ, используемый данным механизмом, должен иметь тип **CKK_GENERIC_SECRET** или **CKK_GOST28147**.

Параметром этого механизма является DER-закодированный идентификатор криптографических параметров ГОСТ Р34.11-94, используемых при вычислении хэш-функции. Параметр механизма может отсутствовать, при этом будут использованы криптографические параметры, определяемые идентификатором *id-GostR3411-94-CryptoProParamSet* (см. RFC 4357 **[5]** раздел 11.2).

Результатом является 32-байтовый вектор, содержащий значение хэш-функции на ключе.

Длины данных для механизма **CKM_GOSTR3411_HMAC**:

Функция	Тип ключа	Длина данных	Длина хэш-вектора

C_Sign	**CKK_GENERIC_SECRET** или **CKK_GOST28147**	Произвольная	32 байта
C_Verify	**CKK_GENERIC_SECRET** или **CKK_GOST28147**	Произвольная	32 байта

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** определяют минимальную и максимальную длину ключа в байтах.

3.3 ГОСТ Р 34.10-2001

3.3.1 Генерация ключевой пары

Генерация ключевой пары ГОСТ Р 34.10-2001 осуществляется с помощью механизма **CKM_GOSTR3410_KEY_PAIR_GEN**.

Этот механизм не имеет параметра.

Механизм создает ключевую пару с криптографическими параметрами, соответствующими атрибутам **CKA_GOSTR3410PARAMS**, **CKA_GOSTR3411PARAMS** и **CKA_GOST28147PARAMS** шаблона ключа проверки. Атрибут **CKA_GOST28147PARAMS** может отсутствовать в шаблоне ключа проверки.

Механизм в дополнение к атрибутам шаблона устанавливает атрибуты **CKA_CLASS**, **CKA_KEY_TYPE**, и **CKA_VALUE** для ключа проверки, и атрибуты **CKA_CLASS**, **CKA_KEY_TYPE**, **CKA_VALUE**, **CKA_GOSTR3410PARAMS**, **CKA_GOSTR3411PARAMS** и **CKA_GOST28147PARAMS** для ключа подписи.

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.3.2 Выработка и проверка ЭЦП

Выработка и проверка ГОСТ Р 34.10-2001 осуществляется при помощи механизма **CKM_GOSTR3410**.

Этот механизм не имеет параметра.

ЭЦП представляет собой 64-байтовый вектор – последовательная запись 32-байтовых векторов s и r' в big-endian представлении (см. RFC 4490 **[6]** раздел 3.2, RFC 4491 **[7]** раздел 2.2.2). Входными данными для механизма является 32-байтовый вектор, содержащий результат вычисления хэш-функции в соответствии с ГОСТ Р 34.11-94 от сообщения/подписанного сообщения, что соответствует режиму обработки single-part в терминологии PKCS#11.

Длины ключей и данных для механизма **CKM_GOST28147_MAC**:

Функция	Тип ключа	Длина данных	Длина ЭЦП
C_Sign[1]	**CKK_GOSTR3410**	32 байта	64 байта
C_Verify[1]	**CKK_GOSTR3410**	32 байта	64 байта

[1] Поддерживается только одношаговая операция, что соответствует режиму обработке single-part в терминологии PKCS#11

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.3.3 Выработка/проверка ЭЦП с предварительной выработкой значения хэш-функции по ГОСТ Р 34.11-94

Выработка и проверка ЭЦП ГОСТ Р 34.10-2001 с предварительной выработкой значения хэш-функции в соответствии с ГОСТ Р 34.11-94 осуществляется при

помощи механизма **CKM_GOSTR3410_WITH_GOSTR3411**.

Параметром этого механизма является DER-закодированный идентификатор криптографических параметров ГОСТ Р 34.11-94, используемых при вычислении хэш-функции. Параметр механизма может отсутствовать, при этом будут использованы криптографические параметры, определяемые значением атрибута **CKA_GOSTR3411PARAMS** используемого ключа.

ЭЦП представляет собой 64-байтовый вектор – последовательная запись 32-байтовых векторов s и r' в big-endian представлении (см. RFC 4490 **[6]** раздел 3.2, RFC 4491 **[7]** раздел 2.2.2). Входными данными для механизма является сообщение/подписанное сообщение, при этом допускается как single-, так и multiple-part обработка в терминологии PKCS#11.

Длины ключей и данных для механизма **CKM_GOST28147_MAC**:

Функция	Тип ключа	Длина данных	Длина ЭЦП
C_Sign	**CKK_GOSTR3410**	Произвольная	64 байта
C_Verify	**CKK_GOSTR3410**	Произвольная	64 байта

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.3.4 Шифрование ключей ГОСТ 28147-89 при помощи ключей ГОСТ Р 34.10-2001

Шифрование симметричных ключей ГОСТ 28147-89 с использованием ключей подписи и проверки ГОСТ Р 34.10-2001 осуществляется при помощи механизма **CKM_GOSTR3410_KEY_WRAP**. Шифрование выполняется в соответствии с алгоритмом, представленным в RFC 4490 **[5]** раздел 5.2. Зашифрованный ключ представлен в виде DER-закодированной структуры, соответствующей ASN.1

типу *GostR3410-KeyTransport*.

Параметром для этого механизма является структура **CK_GOSTR3410_KEY_WRAP_PARAMS**:

```
typedef struct CK_GOSTR3410_KEY_WRAP_PARAMS {
        CK_BYTE_PTR      pWrapOID;
        CK_ULONG         ulWrapOIDLen;
        CK_BYTE_PTR      pUKM;
        CK_ULONG         ulUKMLen;
        CK_OBJECT_HANDLE hKey;
} CK_GOSTR3410_KEY_WRAP_PARAMS;
```

Элементы структуры имеют следующие значения:

pWrapOID	Указатель на буфер, содержащий DER-закодированный идентификатор используемых криптографических параметров ГОСТ 28147-89; при операции **C_WrapKey** если равен NULL_PTR, используются криптографические параметры, определяемые значением атрибута **CKA_GOST28147PARAMS** ключа проверки; при операции **C_UnwrapKey** значение этого указателя не используется и должно быть NULL_PTR
ulWrapOIDLen	Длина в байтах буфера, содержащего DER-закодированный идентификатор используемых криптографических параметров ГОСТ 28147-89
pUKM	Указатель на буфер, содержащий UKM; при операции **C_WrapKey**, если равен NULL_PTR, при вычислениях будет использовано случайное содержимое UKM; при операции **C_UnwrapKey**, если не равен NULL_PTR, будет произведено сравнение содержимого буфера с содержимым UKM ASN.1-структуры зашифрованного ключа и, в случае несовпадения, зашифрованный ключ признается недействительным
ulUKMLen	Длина в байтах буфера содержащего UKM (должна быть равна восьми, если указатель *pUKM* не равен NULL_PTR)
hKey	Дескриптор (handle) ключа отправителя; при операции **C_WrapKey** указывается дескриптор ключа отправителя, при операции **C_UnwrapKey** – дескриптор ключа получателя; если установлен в **CK_INVALID_HANDLE**, генерируется и применяется ключевая пара зашифрователя одноразового использования

При операции **C_UnwrapKey** механизм устанавливает расшифрованный результат как значение атрибута **CKA_VALUE** создаваемого секретного ключа. Все остальные атрибуты должны быть указаны в шаблоне создаваемого ключа.

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.3.5 Выработка общего секретного ключа при помощи ключей ГОСТ Р 34.10-2001

Выработка общего секретного ключа с использованием ключей подписи и проверки ГОСТ Р 34.10-2001 осуществляется с помощью механизма **CKM_GOSTR3410_DERIVE**. Ключ, используемый в механизме, должен иметь класс **CKO_PRIVATE_KEY** и тип **CKK_GOSTR3410**. Выработка осуществляется в соответствии с алгоритмом представленным в RFC 4357 **[5]** раздел 5.2.

Параметром для этого механизма является структура **CK_GOSTR3410_DERIVE_PARAMS**:

```
typedef struct CK_GOSTR3410_DERIVE_PARAMS {
  CK_EC_KDF_TYPE kdf;
  CK_BYTE_PTR    pPublicData;
  CK_ULONG       ulPublicDataLen;
  CK_BYTE_PTR    pUKM;
  CK_ULONG       ulUKMLen;
} CK_GOSTR3410_DERIVE_PARAMS;
```

Элементы структуры имеют следующие значения:

kdf	Идентификатор используемой диверсификации ключа, может принимать

	значения: **CKD_NULL** (диверсификация не производится) или **CKD_CPDIVERSIFY_KDF** (производится диверсификация согласованного ключа в соответствии с алгоритмом, представленным в RFC 4357 **[5]** раздел 6.5).
pPublicData[1]	Указатель на буфер, содержащий ключ проверки получателя.
ulPublicDataLen	Длина в байтах буфера, содержащего ключ проверки получателя (должна равняться 64).
pUKM	Указатель на буфер содержащий UKM.
ulUKMLen	Длина в байтах буфера содержащего UKM (должна равняться 8)

[1] Ключ проверки подписи получателя представлен в виде последовательной записи координат точки X и Y - подряд два вектора длиной по 32 байта в little-endian байтовом порядке, младшие байты сначала.

Используемые при вычислениях криптографические параметры определяются значениями атрибутов ключа подписи **CKA_GOSTR3410PARAMS**, **CKA_GOSTR3411PARAMS** и **CKA_GOST28147PARAMS**.

Механизм устанавливает результат вычислений как значение атрибута **CKA_VALUE** создаваемого секретного ключа. Все остальные атрибуты должны быть указаны в шаблоне создаваемого ключа.

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** не используются.

3.4 Механизмы для использования ГОСТ в TLS 1.0

Механизмы, представленные в настоящем разделе, опираются на публикации **[8]** и **[9]**.

3.4.1 Определения

Механизмы:

CKM_TLS_GOST_PRF

CKM_TLS_GOST_PRE_MASTER_KEY_GEN

CKM_TLS_GOST_MASTER_KEY_DERIVE

CKM_TLS_GOST_KEY_AND_MAC_DERIVE

3.4.2 Параметры механизмов

3.4.2.1 CK_TLS_GOST_PRF_PARAMS; CK_TLS_GOST_PRF_PTR

CK_TLS_GOST_PRF_PARAMS – структура, предоставляющая параметры механизму **CKM_TLS_GOST_PRF**. Определена следующим образом:

```
typedef struct CK_TLS_GOST_PRF_PARAMS {
  CK_TLS_PRF_PARAMS TlsPrfParams;
  CK_BYTE_PTR pHashParamsOid;
  CK_ULONG    ulHashParamsOidLen;
} CK_TLS_GOST_PRF_PARAMS;
```

Поля этой структуры означают:

TlsPrfParams – структура параметров PRF для механизма **CKM_TLS_PRF**. Описана в **[1]** (раздел 12.32.2).

pHashParamsOid – DER-закодированный OID используемых параметров хэш-функции ГОСТ Р 34.11-94.

ulHashParamsOidLen – длина закодированного значения OID.

CK_TLS_GOST_PRF_PARAMS_PTR – указатель на **CK_TLS_GOST_PRF_PARAMS**.

3.4.2.2 CK_TLS_GOST_MASTER_KEY_DERIVE_PARAMS; CK_TLS_GOST_MASTER_KEY_DERIVE_PTR

CK_TLS_GOST_MASTER_KEY_DERIVE_PARAMS – структура, предоставляющая параметры механизму **CKM_TLS_GOST_MASTER_KEY_DERIVE**. Определена следующим образом:

```
typedef struct CK_TLS_GOST_MASTER_KEY_DERIVE_PARAMS {
```

```
  CK_SSL3_RANDOM_DATA RandomInfo;
  CK_BYTE_PTR pHashParamsOid;
  CK_ULONG    ulHashParamsOidLen;
} CK_TLS_GOST_MASTER_KEY_DERIVE_PARAMS;
```

Поля этой структуры означают:

RandomInfo – случайные данные клиента и сервера, структура описана в **[1]** (раздел 12.31.2).

pHashParamsOid – DER-закодированный OID используемых параметров хэш-функции ГОСТ Р 34.11-94.

ulHashParamsOidLen – длина закодированного значения OID.

CK_TLS_GOST_MASTER_KEY_DERIVE_PARAMS_PTR – указатель на **CK_TLS_GOST_MASTER_KEY_DERIVE_PARAMS**.

3.4.2.3 CK_TLS_GOST_KEY_MAT_PARAMS; CK_TLS_GOST_KEY_MAT_PTR

CK_TLS_GOST_KEY_MAT_PARAMS – структура, предоставляющая параметры механизму **CKM_TLS_GOST_KEY_AND_MAC_DERIVE**. Определена следующим образом:

```
typedef struct CK_TLS_GOST_KEY_MAT_PARAMS {
  CK_SSL3_KEY_MAT_PARAMS KeyMatParams;
  CK_BYTE_PTR pHashParamsOid;
  CK_ULONG    ulHashParamsOidLen;
} CK_TLS_GOST_KEY_MAT_PARAMS;
```

Поля этой структуры означают:

KeyMatParams – данные, используемые стандартным механизмом TLS **CKM_TLS_KEY_AND_MAC_DERIVE**. Структура определена в **[1]** (раздел 12.31.2).

pHashParamsOid – DER-закодированный OID используемых параметров хэш-функции ГОСТ Р 34.11-94.

ulHashParamsOidLen – длина закодированного значения OID.

CK_TLS_GOST_KEY_MAT_PARAMS_PTR – указатель на **CK_TLS_GOST_KEY_MAT_PARAMS**.

3.4.3 Генерация псевдослучайной последовательности TLS PRF

Генерация псевдослучайной последовательности при использовании расширения протокола и алгоритмов ГОСТ в TLS 1.0 осуществляется с помощью механизма **CKM_TLS_GOST_PRF**. Он использует обобщенные секретные ключи.

Механизм имеет параметр, структуру **CK_TLS_GOST_PRF_PARAMS**. Этот параметр позволяет передать входные затравку и метку вместе с их длинами, буфер и длину запрашиваемого вывода, а также параметры хэш-функции ГОСТ Р 34.11-94.

Этот механизм вырабатывает псевдослучайный набор байтов указанной длины.

Этот механизм отличается от прочих механизмов выработки ключа тем, что не использует шаблон, передаваемый в функцию **C_DeriveKey**, и, следовательно, этот аргумент должен быть **NULL_PTR**.

Кроме того, механизм возвращает выработанную последовательность не через дескриптор ключа, а через поле **pOutput** структуры **CK_TLS_PRF_PARAMS**, поэтому аргумент **phKey** функции **C_DeriveKey** не используется и должен быть равен **NULL_PTR**.

Если вызов **C_DeriveKey** с этим механизмом завершается неудачей, то считается, что выходные данные не выработаны.

3.4.4 Выработка премастер-ключа

Генерация премастер-ключа при использовании расширения протокола и алгоритмов ГОСТ в TLS 1.0 осуществляется при помощи механизма **CKM_TLS_GOST_PRE_MASTER_KEY_GEN**. Данный механизм генерирует 32-байтовый премастер-ключ, используемый в TLS 1.0 с алгоритмами ГОСТ.

Механизм имеет один параметр, структуру **CK_VERSION**, предоставляющую версию TLS, используемую клиентом.

Механизм устанавливает атрибуты **CKA_CLASS**, **CKA_KEY_TYPE**, **CKA_VALUE** и **CKA_VALUE_LEN** создаваемого объекта ключа. Другие атрибуты могут быть указаны в шаблоне, в противном случае устанавливаются некоторые значения по умолчанию.

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

3.4.5 Выработка мастер-ключа

Выработка мастер-ключа при использовании расширения протокола и алгоритмов ГОСТ в TLS 1.0 осуществляется при помощи механизма **CKM_TLS_GOST_MASTER_KEY_DERIVE**. Он генерирует 48-байтовый обобщенный секретный ключ из 32-байтового обобщенного секретного ключа. Он используется для выработки мастер-ключа протокола TLS из премастер-ключа.

Механизм имеет параметр, структуру **CK_TLS_GOST_MASTER_KEY_DERIVE_PARAMS**, позволяющую передать случайные данные сервера и клиента, а также параметры хэш-функции ГОСТ Р 34.11-94 в функцию **C_DeriveKey**.

Механизм устанавливает атрибуты **CKA_CLASS**, **CKA_KEY_TYPE, CKA_VALUE** и **CKA_VALUE_LEN** создаваемого объекта ключа. Другие атрибуты могут быть указаны в шаблоне, в противном случае им устанавливаются некоторые значения по умолчанию.

Этот механизм имеет следующие правила относительно чувствительности и извлекаемости ключа:

- Атрибуты **CKA_SENSITIVE** и **CKA_EXTRACTABLE** в шаблоне могут быть указаны как любое из значений **CK_TRUE** или **CK_FALSE**. Если они не указаны явно, то им присваивается некоторое значение по умолчанию.
- Если атрибут **CKA_ALWAYS_SENSITIVE** базового ключа был установлен в **CK_FALSE**, то выведенный ключ также будет иметь это значение. Если он был **CK_TRUE**, то выведенный ключ будет иметь значение **CKA_ALWAYS_SENSITIVE**, равное значению **CKA_SENSITIVE**.
- Аналогично, если атрибут **CKA_NEVER_EXTRACTABLE** базового ключа был установлен в **CK_FALSE**, то выведенный ключ также будет иметь это значение. Если он был **CK_TRUE**, то выведенный ключ будет иметь значение **CKA_NEVER_EXTRACTABLE**,противоположное значению **CKA_EXTRACTABLE**.

Для этого механизма поля *ulMinKeySize* и *ulMaxKeySize* структуры **CK_MECHANISM_INFO** равны 48 байтам.

3.4.6 Выработка ключей шифрования и имитозащиты

Выработка ключей шифрования и имитозащиты при использовании расширения протокола и алгоритмов ГОСТ в TLS 1.0 осуществляется с помощью механизма **CKM_TLS_GOST_KEY_AND_MAC_DERIVE**, который вырабатывает ключи шифрования и имитозащиты из мастер-ключа и случайных данных сервера и клиента. Механизм возвращает дескрипторы созданных ключей и инициализационные векторы.

Механизм имеет параметр, структуру **CK_TLS_GOST_KEY_MAT_PARAMS**. Эта структура позволяет передать случайные данные и характеристики ключевого материала для данного шифра TLS, параметры хэш-функции ГОСТ Р 34.11-94, а

также указатели, через которые возвращаются дескрипторы ключей и инициализационные векторы.

Механизм создает четыре ключа класса **CKO_SECRET_KEY**, с типом **CKK_GOST28147**. Прочие атрибуты ключей устанавливаются соответственно шаблону, передаваемому в функцию **C_DeriveKey**. По умолчанию ключи помечаются как пригодные для шифрования, расшифрования и выработки ключей.

Инициализационные векторы вырабатываются и возвращаются, если поле *ulIVSizeInBits* структуры **CK_SSL3_KEY_MAT_PARAMS**, содержащейся в структуре **CK_TLS_GOST_KEY_MAT_PARAMS**, имеет ненулевое значение.

Все четыре ключа наследуют значение атрибутов **CKA_SENSITIVE**, **CKA_ALWAYS_SENSITIVE**, **CKA_EXTRACTABLE** и **CKA_NEVER_EXTRACTABLE** из базового ключа. Шаблон, переданный в функцию **C_DeriveKey**, не может указывать значения этих атрибутов, отличные от значений их у базового ключа.

Следует заметить, что структура **CK_SSL3_KEY_MAT_OUT**, на которую указывает поле *pReturnedKeyMaterial* структуры **CK_SSL3_KEY_MAT_PARAMS**, изменяется функцией **C_DeriveKey**, а именно: поля дескрипторов ключей будут содержать дескрипторы созданных ключей, и буферы, на которые указывают поля *pIVClient* и *pIVServer*, будут заполнены значениями инициализационных векторов (если они были запрошены). Таким образом, эти два поля должны указывать на буферы достаточной длины.

Этот механизм отличается от большинства механизмов выработки ключа тем, какую информацию и как он возвращает. Поскольку механизм возвращает дескрипторы всех созданных им ключей через структуру **CK_SSL3_KEY_MAT_OUT**, аргумент **phKey** функции **C_DeriveKey** не нужен, и должен быть **NULL_PTR**.

Если вызов **C_DeriveKey** завершается неудачей, ни один из четырех ключей не создается.

3.5 Изменения к существующим механизмам

3.5.1 Выработка секретного ключа из пароля с использованием PKCS#5v2 и хэш-функции ГОСТ Р 34.11-94

Выработка секретного ключа из пароля с использованием схемы PKCS#5v2 **[8]** и хеш-функции ГОСТ Р34.11-94 осуществляется с помощью механизма **CKM_PKCS5_PBKD2**. Для этого определяется новое значение типа **CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE**:

Идентификатор	**Тип параметра**
CKP_PKCS5_PBKD2_HMAC_GOSTR3411	Параметром PRF является идентификатор криптографических параметров ГОСТ Р 34.11-94, т.е. *pPrfData* и *ulPrfDataLen* соответственно указатель на буфер и длина в байтах буфера, содержащего DER-закодированный идентификатор используемых криптографических параметров ГОСТ Р 34.11-94. В случае если *pPrfData* имеет значение NULL_PTR, используются криптографические параметры определяемые идентификатором *id-GostR3411-94-CryptoProParamSet* (см. RFC 4357 **[5]** раздел 11.2).

4 Список документов

[1] PKCS #11 v2.20: Cryptographic Token Interface Standard. RSA Laboratories, 28 June 2004.

[2] ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. Издательство стандартов, 1996.

[3] ГОСТ Р 34.10-94. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма. Издательство стандартов, 1994.

[4] ГОСТ Р 34.10-2001. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной подписи. Издательство стандартов, 2001.

[5] RFC 4357, V. Popov, I. Kurepkin, 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”, January 2006.

[6] RFC 4490, S. Leontiev, Ed. G. Chudov, “Using the GOST 28147-89, GOST R 34.11-94,GOST R 34.10-94, and GOST R 34.10-2001 Algorithms with Cryptographic Message Syntax (CMS)”, May 2006.

[7] RFC 4491, S. Leontiev, Ed., D. Shefanovski, Ed., “Using the GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms with the Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, May 2006.

[8] Internet-Draft. Ed. G. Chudov, Ed. S. Leontiev, “GOST 28147-89 Cipher Suites for Transport Layer Security (TLS)”, draft-chudov-cryptopro-cptls-03.txt, Expired March 12, 2007.

[9] RFC 2246, T. Dierks, C. Allen, “The TLS Protocol. Version 1.0”, January 1999.

5 Приложение 1 - Дополнения к заголовочному файлу pkcs11t.h

```
/*
 * NSSCK_VENDOR_???
 * признак VENDOR_DEFINED-значения - взведенный старший бит
 * гарантия уникальности определяемых далее значений - уникальная "база"
 */
#define NSSCK_VENDOR_PKSC11_RU_TEAM  0xc4321000  /* 0x80000000 | 0x54321000 */

/* GOST KEY TYPES */

#define CKK_GOSTR3410   (NSSCK_VENDOR_PKSC11_RU_TEAM |0x000)
#define CKK_GOSTR3411   (NSSCK_VENDOR_PKSC11_RU_TEAM |0x001)
#define CKK_GOST28147   (NSSCK_VENDOR_PKSC11_RU_TEAM |0x002)


/* GOST OBJECT ATTRIBUTES */

#define CKA_GOSTR3410PARAMS    (NSSCK_VENDOR_PKSC11_RU_TEAM |0x001)
#define CKA_GOSTR3411PARAMS    (NSSCK_VENDOR_PKSC11_RU_TEAM |0x002)
#define CKA_GOST28147PARAMS    (NSSCK_VENDOR_PKSC11_RU_TEAM |0x003)

/* GOST MECHANISMS */

#define CKM_GOSTR3410_KEY_PAIR_GEN   (NSSCK_VENDOR_PKSC11_RU_TEAM |0x000)
#define CKM_GOSTR3410                (NSSCK_VENDOR_PKSC11_RU_TEAM |0x001)
#define CKM_GOSTR3410_KEY_WRAP       (NSSCK_VENDOR_PKSC11_RU_TEAM |0x004)
#define CKM_GOSTR3410_DERIVE         (NSSCK_VENDOR_PKSC11_RU_TEAM |0x005)
#define CKM_GOSTR3411                (NSSCK_VENDOR_PKSC11_RU_TEAM |0x020)
#define CKM_GOSTR3411_HMAC           (NSSCK_VENDOR_PKSC11_RU_TEAM |0x021)
#define CKM_GOST28147_KEY_GEN        (NSSCK_VENDOR_PKSC11_RU_TEAM |0x030)
#define CKM_GOST28147_ECB            (NSSCK_VENDOR_PKSC11_RU_TEAM |0x031)
#define CKM_GOST28147                (NSSCK_VENDOR_PKSC11_RU_TEAM |0x032)
#define CKM_GOST28147_MAC            (NSSCK_VENDOR_PKSC11_RU_TEAM |0x033)
#define CKM_PBA_GOSTR3411_WITH_GOSTR3411_HMAC \
                                (NSSCK_VENDOR_PKSC11_RU_TEAM |0x035)
#define CKM_GOST28147_KEY_WRAP       (NSSCK_VENDOR_PKSC11_RU_TEAM |0x036)
#define CKM_TLS_GOST_MASTER_KEY_DERIVE    (NSSCK_VENDOR_PKSC11_RU_TEAM |0x101)
#define CKM_TLS_GOST_KEY_AND_MAC_DERIVE   (NSSCK_VENDOR_PKSC11_RU_TEAM |0x102)
```

```
#define CKM_TLS_GOST_PRF                    (NSSCK_VENDOR_PKSC11_RU_TEAM |0x103)

#define CKD_GOST_KDF   (NSSCK_VENDOR_PKSC11_RU_TEAM |0x001)
#define CKP_PKCS5_PBKD2_HMAC_GOSTR3411 (NSSCK_VENDOR_PKSC11_RU_TEAM |0x001)

typedef struct CK_GOSTR3410_KEY_WRAP_PARAMS {
    CK_BYTE_PTR      pWrapOID;
    CK_ULONG         ulWrapOIDLen;
    CK_BYTE_PTR      pUKM;
    CK_ULONG         ulUKMLen;
    CK_OBJECT_HANDLE hKey;
} CK_GOSTR3410_KEY_WRAP_PARAMS;

typedef CK_GOSTR3410_KEY_WRAP_PARAMS CK_PTR CK_GOSTR3410_KEY_WRAP_PARAMS_PTR;

typedef struct CK_GOST3410_DERIVE_PARAMS {
    CK_BYTE_PTR      pHashOID;
    CK_ULONG         ulHashOIDLen;
    CK_BYTE_PTR      pUKM;
    CK_ULONG         ulUKMLen;
    CK_OBJECT_HANDLE hSenderPrivateKey;
} CK_GOST3410_DERIVE_PARAMS;

typedef CK_GOSTR3410_DERIVE_PARAMS CK_PTR CK_GOSTR3410_DERIVE_PARAMS_PTR;
```

Лист регистрации изменений

Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного документа	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					
