LS11SW2016 – программная реализация PKCS#11 v2.40 с поддержкой ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012, ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 Руководство Программиста

29 ноября 2018 г.

Оглавление

1 Введение **3**
- 1.1 Обзор архитектуры . . . 3
- 1.2 Системные требования . . . 3
- 1.3 Особенности реализации . . . 3

2 Основные сведения **5**
- 2.1 Состав и структура . . . 5
  - 2.1.1 Библиотека ls11sw2016 . . . 5
  - 2.1.2 Поддерживаемые типы объектов . . . 5
  - 2.1.3 Поддерживаемые механизмы . . . 6
- 2.2 Установка . . . 7
- 2.3 Создание программного токена . . . 8

3 Утилиты **10**
- 3.1 Утилита create_sw_token . . . 10
- 3.2 Утилита create_user_license_request . . . 10
- 3.3 Утилита verify_token_license . . . 10
- 3.4 Утилита p11conf . . . 11
  - 3.4.1 Получение информации о библиотеке . . . 12
  - 3.4.2 Получение информации о слотах . . . 12
  - 3.4.3 Получение информации о токене . . . 13
  - 3.4.4 Инициализация токена . . . 13
  - 3.4.5 Назначение PIN администратора безопасности . . . 13
  - 3.4.6 Инициализация пользовательского PIN . . . 14
  - 3.4.7 Изменение пользовательского PIN . . . 15

4 Тестирование **16**
- 4.1 Тестовый проект . . . 16
- 4.2 Запуск тестов . . . 16

5 Ссылки **22**

1 Введение

LS11SW2016 является программной реализацией стандарта PKCS#11 API [10], дополненного поддержкой российских криптографических алгоритмов в соответствии со спецификациями, выработанными Техническим комитетом по стандартизации (ТК 26) "Криптографическая защита информации" [2, 11]. Начиная с версии 6.0, LS11SW2016 поддерживает алгоритмы ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012, ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015, а также сопутствующие алгоритмы и параметры, определенные руководящими документами ТК 26.

Поддержка алгоритмов российской криптографии в LS11SW2016 осуществляется библиотекой с интерфейсом PKCS#11. LS11SW2016 позволяет работать только с одним программным токеном, который располагается в предопределенном месте файловой системы пользователя.

1.1 Обзор архитектуры

LS11SW2016 предоставляет библиотеку PKCS#11, динамически подключаемую приложением. При инициализации библиотеки создается единственный слот для подключения уже созданного программного токена со значением slot ID, равным 0. Программный токен создается в файловой системе пользователя утилитой `create_sw_token`.

LS11SW2016 API включает функции, описанные в спецификации PKCS#11 API [10], с учетом расширения алгоритмами российской криптографии [11, 12, 13].

1.2 Системные требования

LS11SW2016 реализована кросс-платформенным образом, так что она, в принципе, может быть портирована в любую операционную систему, где поддерживается язык Си. Текущая версия работает в операционных системах Windows, Linux и Mac OS X на 32-х и 64-х разрядных платформах. Имеется также вариант библиотеки для мобильных устройств с операционной системой Android.

1.3 Особенности реализации

В LS11SW2016 поддерживаются все российские криптографические механизмы, определенные в [11, 12, 13]. Все криптографические механизмы поддерживаются библиотекой программно. Кроме того, библиотека позволяет работать с временными сессионными объектами для всех поддерживаемых механизмов.

В интересах поддержки российской реализации стандарта PKCS#12 дополнительно реализованы некоторые дополнительные механизмы. Например, добавлен механизм `CKM_PBA_GOSTR3411_WITH_GOSTR3411_HMAC`, используемый для генерации ключа на пароле при выработке и проверке дайджеста хранилища. Добавлен механизм `CKM_GOST28147_PKCS8_KEY_WRAP` для упаковки и распаковки закрытого ключа ГОСТ Р 34.10-2001 по стандарту PKCS#8 с помощью шифрования секретным ключом, генерируемом на пароле механизмом `CKM_PKCS5_PBKD2`. С порядком использования всех механизмов библиотеки можно ознакомиться в тестовом проекте `ls11sw_sdk`, который можно скачать с сайта ООО "ЛИССИ-Софт".

Если при генерации ключевой пары в шаблоне закрытого ключа не был задан атрибут `CKA_ID`, то он автоматически устанавливается равным SHA-1 от значения открытого ключа в соответствии с мировой практикой. Такой способ позволяет установить связь между объектами ключевой пары на токене.

Функции генерации случайных чисел поддерживаются встроенным в библиотеку криптографическим генератором. Библиотека использует начальное значение для инициализации генератора (40 байтов) из предопределенного файла `prng_start.bin`, который располагается в папке программного токена. Путь к папке программного токена:

```
в Windows:
   %USERPROFILE%\.LS11SW2016
в Linux:
   $HOME/.LS11SW2016
```

Программный токен с файлом `prng_start.bin` создается специальной утилитой командной строки `create_sw_token`, поэтому она запускается из инсталлятора. Содержимое данного файла изменяется при каждой инициализации библиотеки функцией `C_Initialize()`, поэтому при следующем запуске любого приложения, использующего библиотеку, ДСЧ будет инициализирован другим значением.

Библиотека поддерживает работу из нескольких потоков, однако по стандарту нельзя использовать одну и ту же сессию из разных потоков.

Количество одновременно открытых сессий в приложении не должно превышать 256.

Создание и лицензирование токена производится через web-интерфейс на сайте "ЛИССИ-Софт"[1]. После лицензирования на токене нужно установить метку и PIN пользователя. Его дальнейшую инициализацию с присвоением метки и установку PIN можно выполнить либо утилитой p11conf, либо программно.

Программный токен считается не извлекаемым, поэтому никакие события на слоте библиотекой не регистрируются.

2 Основные сведения

2.1 Состав и структура

В состав инсталлятора LS11SW2016 входят:

- Библиотека PKCS#11 ls11sw2016
- Утилита создания программного токена `create_sw_token`
- Утилита создания файла запроса на лицензию `create_user_license_request`
- Утилита проверки состояния лицензионного файла `verify_token_license`
- Утилита конфигурирования токена p11conf.

Данное Руководство программиста и SDK с тестовыми примерами можно скачать отдельно с сайта ООО "ЛИССИ-Софт".

Имена файлов динамических библиотек и выполняемых файлов в разных операционных системах отличаются. Например, в Windows имя файла библиотеки ls11sw2016 - ls11sw2016.dll, в Linux - libls11sw2016.so, в Mac OS X - libls11sw2016.dylib. Это отличие нужно учитывать при загрузке библиотеки из прикладной программы. Например, выполняемый файл утилиты p11conf в Windows называется p11conf.exe, а в Linux и Mac OS X - просто p11conf.

2.1.1 Библиотека ls11sw2016

Библиотека PKCS#11 API ls11sw2016 должна быть загружена прикладной программой динамически, после чего из нее извлекается список функций PKCS#11.

Затем вызывается функция `C_Initialize`, с которой начинается работа с библиотекой по стандарту PKCS#11.

2.1.2 Поддерживаемые типы объектов

Программный токен LS11SW2016 в слоте 0 может хранить объекты ключевой пары (`CKO_PRIVATE_KEY`, `CKO_PUBLIC_KEY`) типа `CKK_GOSTR3410` или `CKK_GOSTR3410_512`, объекты сертификатов `CKO_CERTIFICATE` типа X.509, объекты данных `CKO_DATA`, ключи для симметричного шифрования и хеширования, а также параметры домена.

На том же нулевом слоте могут быть созданы временные сессионные объекты перечисленных типов.

2.1.3 Поддерживаемые механизмы

LS11SW2016 поддерживает следующие механизмы PKCS#11:

```
CKM_GOSTR3410_KEY_PAIR_GEN (0x1200)
CKM_GOSTR3410_512_KEY_PAIR_GEN (0xD4321005)
CKM_GOSTR3410 (0x1201)
CKM_GOSTR3410_512 (0xD4321006)
CKM_GOSTR3410_WITH_GOSTR3411 (0x1202)
CKM_GOSTR3410_WITH_GOSTR3411_12_256 (0xD4321008)
CKM_GOSTR3410_WITH_GOSTR3411_12_512 (0xD4321009)
CKM_GOSTR3410_DERIVE (0x1204)
CKM_GOSTR3410_12_DERIVE (0xD4321007)
CKM_GOSTR3410_2012_VKO_256 (0xD4321045)
CKM_GOSTR3410_2012_VKO_512 (0xD4321046)
CKM_KDF_4357 (0xD4321025)
CKM_KDF_GOSTR3411_2012_256 (0xD4321026)
CKM_KDF_TREE_GOSTR3411_2012_256 (0xD4321044)
CKM_GOSTR3410_KEY_WRAP (0x1203)
CKM_GOSTR3410_PUBLIC_KEY_DERIVE (0xD432100A)
CKM_LISSI_GOSTR3410_PUBLIC_KEY_DERIVE (0xD4321037)
CKM_GOST_GENERIC_SECRET_KEY_GEN (0xD4321049)
CKM_GOST_CIPHER_KEY_GEN (0xD4321048)
CKM_GOST_CIPHER_ECB (0xD4321050)
CKM_GOST_CIPHER_CBC (0xD4321051)
CKM_GOST_CIPHER_CTR (0xD4321052)
CKM_GOST_CIPHER_OFB (0xD4321053)
CKM_GOST_CIPHER_CFB (0xD4321054)
CKM_GOST_CIPHER_OMAC (0xD4321055)
CKM_GOST_CIPHER_KEY_WRAP (0xD4321059)
CKM_GOST_CIPHER_ACPKM_CTR (0xD4321057)
CKM_GOST_CIPHER_ACPKM_OMAC (0xD4321058)
CKM_GOST28147_KEY_GEN (0x1220)
CKM_GOST28147 (0x1222)
CKM_GOST28147_KEY_WRAP (0x1224)
CKM_GOST28147_PKCS8_KEY_WRAP (0xD4321036)
CKM_GOST_CIPHER_PKCS8_KEY_WRAP (0xD432105A)
CKM_GOST28147_ECB (0x1221)
CKM_GOST28147_CNT (0xD4321825)
CKM_GOST28147_MAC (0x1223)
CKM_KUZNYECHIK_KEY_GEN (0xD4321019)
CKM_KUZNYECHIK_ECB (0xD432101A)
CKM_KUZNYECHIK_CBC (0xD432101E)
CKM_KUZNYECHIK_CTR (0xD432101B)
CKM_KUZNYECHIK_OFB (0xD432101D)
```

```
CKM_KUZNYECHIK_CFB (0xD432101C)
CKM_KUZNYECHIK_OMAC (0xD432101F)
CKM_KUZNYECHIK_KEY_WRAP (0xD4321028)
CKM_KUZNYECHIK_ACPKM_CTR (0xD4321042)
CKM_KUZNYECHIK_ACPKM_OMAC (0xD4321043)
CKM_MAGMA_KEY_GEN (0xD432102A)
CKM_MAGMA_ECB (0xD4321018)
CKM_MAGMA_CBC (0xD4321023)
CKM_MAGMA_CTR (0xD4321020)
CKM_MAGMA_OFB (0xD4321022)
CKM_MAGMA_CFB (0xD4321021)
CKM_MAGMA_OMAC (0xD4321024)
CKM_MAGMA_KEY_WRAP (0xD4321029)
CKM_MAGMA_ACPKM_CTR (0xD4321040)
CKM_MAGMA_ACPKM_OMAC (0xD4321041)
CKM_GOSTR3411 (0x1210)
CKM_GOSTR3411_12_256 (0xD4321012)
CKM_GOSTR3411_12_512 (0xD4321013)
CKM_GOSTR3411_HMAC (0x1211)
CKM_GOSTR3411_12_256_HMAC (0xD4321014)
CKM_GOSTR3411_12_512_HMAC (0xD4321015)
CKM_PKCS5_PBKD2 (0x3B0)
CKM_PBA_GOSTR3411_WITH_GOSTR3411_HMAC (0xD4321035)
CKM_TLS_GOST_KEY_AND_MAC_DERIVE (0xD4321033)
CKM_TLS_GOST_PRE_MASTER_KEY_GEN (0xD4321031)
CKM_TLS_GOST_MASTER_KEY_DERIVE (0xD4321032)
CKM_TLS_GOST_PRF (0xD4321030)
CKM_TLS_GOST_PRF_2012_256 (0xD4321016)
CKM_TLS_GOST_PRF_2012_512 (0xD4321017)
CKM_TLS12_MASTER_KEY_DERIVE (0x3E0)
CKM_TLS12_KEY_AND_MAC_DERIVE (0x3E1)
CKM_TLS_MAC (0x3E4)
CKM_TLS_KDF (0x3E5)
CKM_TLS_TREE_GOSTR3411_2012_256 (0xD4321047)
CKM_EXTRACT_KEY_FROM_KEY (0x365)
CKM_SHA_1 (0x220)
CKM_MD5 (0x210)
```

2.2 Установка

Библиотека ls11sw2016 устанавливается инсталлятором в целевую папку в соответствии с правилами ОС. Кроме того, инсталлятором устанавливаются упомянутые выше утилиты. Для установки пользователю требуются полномочия администрато-

ра. Установка всегда должна производиться лично пользователем в своем сеансе, потому что создаваемый при этом программный токен – это хранилище исключительно приватного использования. В Windows подтверждение полномочий администратора будет запрошено при установке. В Linux инсталлятор должен запускаться командой sudo, например в Linux x86_64:

```
>sudo bash ls11sw-6.9.0-install-linux-x86_64.sh
```

Разрешение на выполнение команды sudo обычно обеспечивается в Linux включением пользователя в группу wheel.

В конце установки запускается утилита `create_sw_token`, которая создает папку программного токена и в диалоге с оператором генерирует файл начального значения для ДСЧ библиотеки. Без этого файла библиотека работать не будет.

Если файл начального значения ДСЧ не создан, то функция инициализации библиотеки ls11sw2016 вернет сообщение об ошибке и завершится с кодом `CKR_FUNCTION_FAILED`:

```
C_Initialize: Can't open random seed file C:\Documents and Settings\
All Users\Application Data\LISSI-Soft\prng_start.bin
C_Initialize error: 6 ( CKR_FUNCTION_FAILED)
..\..\src\src\api_interface.c:1525 API not initialized
```

Важное замечание. На целевой платформе нужно обеспечить прикладным программам возможность найти библиотеки и утилиты по именам путем добавления полного пути к содержащей их папке к значению переменной среды `PATH` или `LD_LIBRARY_PATH`, в зависимости от операционной системы.

Документация и тестовые примеры для LS11SW2016 скачиваются отдельно с сайта "ЛИССИ-Софт"[1].

2.3 Создание программного токена

Программный токен создается исталлятором в предопределенном месте файловой системы пользователя. Этому программному токену присваивается уникальный серийный номер, под которым он регистрируется на сервере в соответствии с его лицензией. С данным программным токеном сможет работать только тот пользователь, который его создавал, причем только на том же компьютере и в той же операционной системе. Для получения лицензии нужно создать файл запроса LIC.REQ утилитой `create_user_license_request` и передать на утверждение на сайте ООО "ЛИССИ-Софт" [1]. После утверждения нужно скачать с сайта файл лицензии LIC.DAT и сохранить его в папке токена.

Папка программного токена называется `.LS11SW2016` и размещается в домашней папке пользователя:

- Windows: `%USERPROFILE%\.LS11SW2016`
- Unix, пользователь: `$HOME/.LS11SW2016`

Важное замечание: Для отображения файлов, имена которых начинаются с точки, может понадобиться указать в настройках файлового менеджера разрешение показа скрытых или системных файлов.

Программный токен первоначально создается инициализированным, но без установленного PIN пользователя. Инициализация токена с присвоением метки и назначение PIN пользователя могут быть выполнены с помощью утилиты конфигурации p11conf (см. раздел "Утилита конфигурации") или программно функциями PKCS#11. Для созданного токена устанавливается умалчиваемое значение PIN администратора безопасности (Security Officer или сокращенно SO) – 87654321.

Заметим, что для доступа к программному токену в файловом пространстве пользователя SO должен работать в сеансе пользователя токена. Фактически это означает, что для программного токена LS11SW2016 SO и пользователь токена – это одно и то же лицо, хотя значения PIN у SO и пользователя могут быть разными.

Если по какой-то причине файлы программного токена оказались поврежденными, то папку программного токена можно удалить и провести лицензирование заново. Поскольку в базе данных сервера уже имеется запись о предыдущем лицензировании, то повторное лицензирование не потребует оплаты.

Программный токен – это просто папка с файлами с точки зрения операционной системы. Поэтому пользователь может в любой момент создать резервную копию лицензированного программного токена и хранить ее на любом носителе для последующего восстановления, если понадобится. Однако, такая копия будет работать только на том же компьютере и у того же пользователя, который производил лицензирование.

3 Утилиты

3.1 Утилита create_sw_token

Обычно вызов утилиты производится без аргументов. Аргумент auto_init_rng носит технологический характер и предназначен только для тестового использования при разработке ПО.

```
Usage: create_sw_token [auto_init_rng]
        auto_init_rng - automatic initial value generation
                for cryptographic random bytes generator
                (optionally used for development needs only).
  Copyright (C) LISSI-Soft, 2016-2018
```

3.2 Утилита create_user_license_request

Данная утилита используется для создания файла запроса на лицензию продукта, заданного параметром model. Файл называется LIC.REQ и создается в текущей папке вызова утилиты.

```
Usage: create_user_license_request model [code]
        model - product name (case sensitive)
        code - license activation code (case sensitive)
  Copyright (C) LISSI-Soft, 2016-2018
```

Для создания запроса на лицензию программного токена LS11SW2016 утилиту нужно вызвать следующим образом:

```
>create_user_license_request LS11SW2016
```

Файл LIC.REQ предается на сервер лицензирования ООО "ЛИССИ-Софт" для получения лицензионного файла LIC.DAT. Файл LIC.DAT нужно скопировать в папку программного токена.

3.3 Утилита verify_token_license

Данная утилита позволяет установить состояние лицензии программного токена.

```
Usage: verify_token_license
  Copyright (C) LISSI-Soft, 2018
```

Пример использования:

```
>verify_token_license
Verifying token license...
Token licensed for vblazhnov until 22.11.2019
License OK
```

3.4 Утилита p11conf

Утилита командной строки p11conf для конфигурирования и администрирования токенов работает исключительно через API, поэтому ее можно применять для работы с любыми библиотеками PKCS#11, а не только с ls11sw2016.

Заметим, что с флагом -c задается идентификатор слота. Этот идентификатор необходимо задавать, когда операция выполняется с конкретным токеном. Идентификаторы всех слотов можно получить с флагом -s.

Утилита предоставляет возможности, о которых сообщается при запуске команды `p11conf -h`. К ним относятся инициализация токена, инициализация и изменение PIN администратора безопасности, инициализация и изменение PIN пользователя и др. Следующие примеры показывают опции утилиты p11conf и выдачу информации о слотах в системе до инициализации токенов.

Важное замечание: Не рекомендуется использовать для "боевых" значений PIN только цифровые символы – это существенно облегчает злоумышленникам подбор PIN. Лучше всего использовать и цифры, и буквы разного размера, и служебные символы. Кроме того, чем длиннее значение PIN, тем сложнее его подобрать. Разрешается использовать любые символы в кодировке UTF-8. Однако, не рекомендуется использовать русские буквы, во избежание путаницы с кодировками в программах ввода.

```
>p11conf -h
Usage:
p11conf [-hitsmIupPredf] -A <PKCS#11 library path>
        [-c <slot ID> -U <user PIN> -S <SO PIN> -n <new PIN> -L <label>]
Flags:
        -h display usage
        -i display PKCS#11 library info
        -s display slot(s) info (-c <slot ID> is optional)
        -t display token(s) info (-c <slot ID> is optional)
Others must use -c <slot ID> too
        -m display mechanism list
        -I initialize token
        -u initialize user PIN
        -p set the user PIN
        -P set the SO PIN
        -e enumerate objects
```

```
        -d dump all object attributes (additional to -e and to -f)
        -r remove all objects
        -e -r remove enumerated objects with prompt
        -f enumerate certificates and write them to DER-files with prompt
Version 5.7
Copyright(C) LISSI-Soft Ltd (http://soft.lissi.ru) 2011-2018
```

Важное замечание: Для библиотеки ls11sw2016 значение slot ID должно задаваться равным 0, так как она работает с единственным неизвлекаемым программным токеном, подключенным к нулевому слоту. У других библиотек может быть несколько слотов для подключения токенов со своими специфическими значаниями идентификаторов.

3.4.1 Получение информации о библиотеке

```
>p11conf -A ls11sw2016 -i
PKCS#11 Info
        Version 2.40
        Manufacturer: LISSI-Soft, Ltd
        Flags: 0x0
        Library Description: ls11sw2016 PKCS#11 library
        Library Version 6.9
OK
```

Данная программа также позволяет выполнять некоторые простые запросы слотам и токенам, например для получения информации о слотах, токенах и для получения списка поддерживаемых механизмов.

Если файл начального значения ДСЧ не создан утилитой `create_sw_token`, то функция инициализации библиотеки ls11sw2016 вернет сообщение об ошибке:

```
>p11conf -A ls11sw2016 -i
C_Initialize: Can't open random seed file C:\Documents and Settings\
All Users\Application Data\LISSI-Soft\prng_start.bin
C_Initialize error: 6 ( CKR_FUNCTION_FAILED)
..\..\src\src\api_interface.c:1525 API not initialized
```

3.4.2 Получение информации о слотах

```
>p11conf -A ls11sw2016 -s
Slot ID 0 Info
        Description: LS11SW2016 Slot 0
        Manufacturer: LISSI-Soft
        Flags: 0x7 (TOKEN_PRESENT)
        Hardware Version: 2.0
        Firmware Version: 2.0
OK
```

3.4.3 Получение информации о токене

```
>p11conf -A ls11sw2016 -t -c 0
Token #0 Info:
        Label: Token Label
        Manufacturer: LISSI-Soft
        Model: LS11SW2016
        Serial Number: 1234567887654321
        Flags: 0x40C (LOGIN_REQUIRED|USER_PIN_INITIALIZED|TOKEN_INITIALIZED)
        Sessions: 0/256
        R/W Sessions: 0/256
        PIN Length: 4-32
        Public Memory: 0xFFFFFFFF/0xFFFFFFFF
        Private Memory: 0xFFFFFFFF/0xFFFFFFFF
        Hardware Version: 1.0
        Firmware Version: 1.0
        Time: 18:58:42
OK
```

Программный токен создается инициализированным, с умалчиваемым значением SO PIN: 87654321 . Прежде чем приложение сможет использовать новый токен LS11SW2016, нужно выполнить следующие начальные шаги (каждому шагу соответствует пример кода):

3.4.4 Инициализация токена

Перед использованием токена он должен быть однажды инициализирован. Ключевой частью данного процесса является назначение токену уникальной метки (имени). Для этого понадобится PIN администратора безопасности (SO) для данного токена (начальное значение SO PIN для только что созданного токена LS11SW2016 – 87654321).

Замечание. Все значения PIN отображаются звездочками при вводе.

```
>p11conf -A ls11sw2016 -I -c 0
Enter the SO PIN: 87654321
Enter a unique token label: LissiSW
```

То же самое можно выполнить, задавая значения SO PIN и метки прямо в командной строке:

```
>p11conf -A ls11sw2016 -I -c 0 -S 87654321 -L LissiSW
```

3.4.5 Назначение PIN администратора безопасности

Правильной организационной практикой является изменение администратором безопасности своего PIN сразу после инициализации токена. Данная процедура предот-

вращает возможность инициализировать токен посторонним лицам и удалить тем самым все созданные объекты (например, ключи и сертификаты).

```
>p11conf -A ls11sw2016 -P -c 0
Enter the SO PIN: 87654321
Enter the new SO PIN: 76543210
Re-enter the new SO PIN: 76543210
```

Вариант ввода в командной строке:

```
>p11conf -A ls11sw2016 -P -c 0 -S 87654321 -n 76543210
```

PIN не обязательно должен быть цифровым, допускаются любые символы, но во избежание проблем с кодировками рекомендуется использовать латинскую половину кодовой таблицы.

3.4.6 Инициализация пользовательского PIN

Данная операция выполняется администратором безопасности после инициализации токена перед передачей его пользователю. Программный токен изначально создается в файловом пространстве пользователя, однако формальные требования стандарта должны быть выполнены и для него.

```
>p11conf -A ls11sw2016 -u -c 0
Enter the SO PIN: 76543210
Enter the new user PIN: 12345678
Re-enter the new user PIN: 12345678
```

Вариант ввода в командной строке:

```
>p11conf -A ls11sw2016 -u -c 0 -S 76543210 -n 12345678
```

Заметим, что попытка повторно выполнить инициализацию пользовательского PIN завершится с ошибкой, потому что данная операция может быть выполнена только сразу после инициализации токена перед передачей его пользователю.

Важное замечание: После трех подряд попыток ввода неправильного пользовательского PIN токен блокирует логин пользователя. Если легальный пользователь сделал это случайно и знает правильное значение PIN, то у него еще есть возможность разблокировать токен. В библиотеке ls11sw2016 имеется возможность разблокировки такого токена с помощью функции инициализации пользовательского PIN от имени SO. Если токен блокировал логин пользователя, то SO и пользователь могут вместе разблокировать его утилитой p11conf с флагом -u. При этом, в качестве значения `new user PIN` нужно дважды ввести известное пользователю правильное значение PIN:

```
...
>p11conf -A ls11sw2016 -e -c 0
User PIN locked

>p11conf -A ls11sw2016 -u -c 0
Enter the SO PIN: ********
Enter the new user PIN: ********
Re-enter the new user PIN: ********
OK

>p11conf -A ls11sw2016 -e -c 0
Enter user PIN: ********
Token objects:
1: CKO_PRIVATE_KEY
         label: 'pki'
2: CKO_DATA
         label: 'data1527'
3: CKO_PUBLIC_KEY
         label: 'pki'
4: CKO_CERTIFICATE
         label: 'pki'
OK
```

Если SO и пользователь – это разные лица, то при таком способе разблокировки и SO, и пользователь не должны показывать друг-другу значения своих PIN при вводе.

3.4.7 Изменение пользовательского PIN

Первое, что должен сделать пользователь после получения токена от администратора безопасности, – это изменение PIN.

```
>p11conf -A ls11sw2016 -p -c 0
Enter user PIN: 12345678
Enter the new user PIN: 01234567
Re-enter the new user PIN: 01234567
```

Вариант ввода в командной строке:

```
>p11conf -A ls11sw2016 -p -c 0 -U 12345678 -n 01234567
```

Если значение SO PIN для токена потеряно, а токен заблокирован из-за нескольких вводов неверного PIN, то штатными средствами LS11SW2016 разблокировать токен нельзя из соображений секретности его данных — его можно только создать заново.

Замечание. Начальное значение USER PIN, установленное SO, запрещается в дальнейшем устанавливать пользователю из соображений безопасности.

4 Тестирование

Пользователям предоставляется тестовый CMake-проект`ls11sw_sdk`, содержащий программы, проверяющие функционирование различных механизмов PKCS#11. Эти программы могут также служить примерами для разработки собственных прикладных программ. Для генерации проектных файлов в операционной системе должна быть установлена сборочная система CMake. Для запуска тестовых примеров на выполнение нужно предварительно установить библиотеку ls11sw2016 соответствующим инсталлятором.

4.1 Тестовый проект

Для генерации проектных файлов нужно из папки build вызвать команду:

```
>cmake ..
```

В Windows может понадобиться дополнительно указать генератор проекта для конкретной версии MS Visual Studio и разрядности выполняемых файлов, соответствующих инсталлятору. Например, для 64-хразрядных файлов:

```
>cmake .. -G "Visual Studio 9 2008 Win64"
```

Для 32-х разрядных файлов:

```
>cmake .. -G "Visual Studio 9 2008"
```

В результате, в системе будут созданы проектные файлы для имеющейся среды программирования. Далее сборка тестов производится либо средствами MS Visual Studio (в Windows), либо командой make (в Linux). В Windows проектный файл для MS Visual Studio будет называться `pkcs11_tests.sln`.

4.2 Запуск тестов

В программах тестового проекта по умолчанию предполагается, что у пользовательского токена SO PIN равен 76543210, а USER PIN равен 01234567. При необходимости, эти значения можно изменить в сборочном файле проекта CMakeLists.txt.

При запуске команды ctest из папки build будут поочередно запущены все тесты. Названия большинства тестов указывают либо на название тестируемого механизма, либо на тестируемый класс объекта.

```
Test project G:/ls11sw_tests/build
      Start  1: info
 1/87 Test  #1: info ................................   Passed    0.97 sec
      Start  2: cko_data
 2/87 Test  #2: cko_data ............................   Passed    1.50 sec
      Start  3: cko_certificate
 3/87 Test  #3: cko_certificate .....................   Passed    1.74 sec
      Start  4: ckm_gostr3411
 4/87 Test  #4: ckm_gostr3411 .......................   Passed    1.00 sec
      Start  5: ckm_gostr3411_12_256
 5/87 Test  #5: ckm_gostr3411_12_256 ................   Passed    1.01 sec
      Start  6: ckm_gostr3411_12_512
 6/87 Test  #6: ckm_gostr3411_12_512 ................   Passed    0.97 sec
      Start  7: ckm_gostr3411_hmac
 7/87 Test  #7: ckm_gostr3411_hmac ..................   Passed    0.93 sec
      Start  8: ckm_gostr3411_12_256_hmac
 8/87 Test  #8: ckm_gostr3411_12_256_hmac ...........   Passed    0.92 sec
      Start  9: ckm_gostr3411_12_512_hmac
 9/87 Test  #9: ckm_gostr3411_12_512_hmac ...........   Passed    0.99 sec
      Start 10: ckm_kdf_gostr3411_2012_256
10/87 Test #10: ckm_kdf_gostr3411_2012_256 ..........   Passed    0.91 sec
      Start 11: ckm_kdf_tree_gostr3411_2012_256
11/87 Test #11: ckm_kdf_tree_gostr3411_2012_256 .....   Passed    0.96 sec
      Start 12: ckm_gost_generic_secret_key_gen
12/87 Test #12: ckm_gost_generic_secret_key_gen .....   Passed    0.99 sec
      Start 13: ckm_extract_key_from_key
13/87 Test #13: ckm_extract_key_from_key ............   Passed    0.92 sec
      Start 14: ckm_gost_cipher_key_gen
14/87 Test #14: ckm_gost_cipher_key_gen .............   Passed    0.99 sec
      Start 15: ckm_gost_cipher_ecb
15/87 Test #15: ckm_gost_cipher_ecb .................   Passed    1.00 sec
      Start 16: ckm_gost_cipher_cbc
16/87 Test #16: ckm_gost_cipher_cbc .................   Passed    0.89 sec
      Start 17: ckm_gost_cipher_ctr
17/87 Test #17: ckm_gost_cipher_ctr .................   Passed    0.86 sec
      Start 18: ckm_gost_cipher_ofb
18/87 Test #18: ckm_gost_cipher_ofb .................   Passed    0.98 sec
      Start 19: ckm_gost_cipher_cfb
19/87 Test #19: ckm_gost_cipher_cfb .................   Passed    0.89 sec
      Start 20: ckm_gost_cipher_acpkm_ctr
20/87 Test #20: ckm_gost_cipher_acpkm_ctr ...........   Passed    0.94 sec
      Start 21: ckm_gost_cipher_omac
21/87 Test #21: ckm_gost_cipher_omac ................   Passed    1.08 sec
      Start 22: ckm_gost_cipher_acpkm_omac
```

```
22/87 Test #22: ckm_gost_cipher_acpkm_omac ..............   Passed    1.00 sec
      Start 23: ckm_gost_cipher_key_wrap
23/87 Test #23: ckm_gost_cipher_key_wrap ................   Passed    0.98 sec
      Start 24: ckm_gost_cipher_pkcs8_key_wrap
24/87 Test #24: ckm_gost_cipher_pkcs8_key_wrap ..........   Passed    3.02 sec
      Start 25: ckm_gost28147_key_gen
25/87 Test #25: ckm_gost28147_key_gen ...................   Passed    0.94 sec
      Start 26: ckm_gost28147
26/87 Test #26: ckm_gost28147 ...........................   Passed    1.00 sec
      Start 27: ckm_gost28147_ecb
27/87 Test #27: ckm_gost28147_ecb .......................   Passed    0.94 sec
      Start 28: ckm_gost28147_ecb_mac_wrap
28/87 Test #28: ckm_gost28147_ecb_mac_wrap ..............   Passed    0.97 sec
      Start 29: ckm_gost28147_cnt
29/87 Test #29: ckm_gost28147_cnt .......................   Passed    1.05 sec
      Start 30: ckm_gost28147_cbc
30/87 Test #30: ckm_gost28147_cbc .......................   Passed    1.08 sec
      Start 31: ckm_gost28147_mac
31/87 Test #31: ckm_gost28147_mac .......................   Passed    1.05 sec
      Start 32: ckm_gost28147_cfb_random
32/87 Test #32: ckm_gost28147_cfb_random ................   Passed    1.16 sec
      Start 33: ckm_gost28147_key_wrap
33/87 Test #33: ckm_gost28147_key_wrap ..................   Passed    1.08 sec
      Start 34: ckm_gost28147_pkcs8_key_wrap
34/87 Test #34: ckm_gost28147_pkcs8_key_wrap ............   Passed    4.78 sec
      Start 35: ckm_kdf_4357
35/87 Test #35: ckm_kdf_4357 ............................   Passed    0.92 sec
      Start 36: ckm_magma_key_gen
36/87 Test #36: ckm_magma_key_gen .......................   Passed    1.18 sec
      Start 37: ckm_magma_ecb
37/87 Test #37: ckm_magma_ecb ...........................   Passed    0.94 sec
      Start 38: ckm_magma_cbc
38/87 Test #38: ckm_magma_cbc ...........................   Passed    0.88 sec
      Start 39: ckm_magma_ctr
39/87 Test #39: ckm_magma_ctr ...........................   Passed    0.95 sec
      Start 40: ckm_magma_acpkm_ctr
40/87 Test #40: ckm_magma_acpkm_ctr .....................   Passed    0.98 sec
      Start 41: ckm_magma_ofb
41/87 Test #41: ckm_magma_ofb ...........................   Passed    0.99 sec
      Start 42: ckm_magma_cfb
42/87 Test #42: ckm_magma_cfb ...........................   Passed    0.83 sec
      Start 43: ckm_magma_omac
43/87 Test #43: ckm_magma_omac ..........................   Passed    0.88 sec
      Start 44: ckm_magma_acpkm_omac
```

```
44/87 Test #44: ckm_magma_acpkm_omac ..................   Passed    0.87 sec
      Start 45: ckm_magma_key_wrap
45/87 Test #45: ckm_magma_key_wrap .....................   Passed    0.88 sec
      Start 46: ckm_magma_cfb_errors
46/87 Test #46: ckm_magma_cfb_errors ...................   Passed    0.96 sec
      Start 47: ckm_magma_cfb_random
47/87 Test #47: ckm_magma_cfb_random ...................   Passed    0.93 sec
      Start 48: ckm_kuznyechik_key_gen
48/87 Test #48: ckm_kuznyechik_key_gen .................   Passed    1.05 sec
      Start 49: ckm_kuznyechik_ecb
49/87 Test #49: ckm_kuznyechik_ecb .....................   Passed    0.95 sec
      Start 50: ckm_kuznyechik_cbc
50/87 Test #50: ckm_kuznyechik_cbc .....................   Passed    0.94 sec
      Start 51: ckm_kuznyechik_ctr
51/87 Test #51: ckm_kuznyechik_ctr .....................   Passed    0.97 sec
      Start 52: ckm_kuznyechik_acpkm_ctr
52/87 Test #52: ckm_kuznyechik_acpkm_ctr ...............   Passed    0.83 sec
      Start 53: ckm_kuznyechik_ofb
53/87 Test #53: ckm_kuznyechik_ofb .....................   Passed    0.92 sec
      Start 54: ckm_kuznyechik_cfb
54/87 Test #54: ckm_kuznyechik_cfb .....................   Passed    0.99 sec
      Start 55: ckm_kuznyechik_omac
55/87 Test #55: ckm_kuznyechik_omac ....................   Passed    0.95 sec
      Start 56: ckm_kuznyechik_acpkm_omac
56/87 Test #56: ckm_kuznyechik_acpkm_omac ..............   Passed    0.83 sec
      Start 57: ckm_kuznyechik_key_wrap
57/87 Test #57: ckm_kuznyechik_key_wrap ................   Passed    0.86 sec
      Start 58: ckm_kuznyechik_cfb_random
58/87 Test #58: ckm_kuznyechik_cfb_random ..............   Passed    0.95 sec
      Start 59: ckm_gostr3410_key_pair_gen
59/87 Test #59: ckm_gostr3410_key_pair_gen .............   Passed    1.54 sec
      Start 60: ckm_gostr3410_public_key_derive
60/87 Test #60: ckm_gostr3410_public_key_derive .........   Passed    2.49 sec
      Start 61: ckm_gostr3410
61/87 Test #61: ckm_gostr3410 ..........................   Passed    1.58 sec
      Start 62: ckm_gostr3410_512
62/87 Test #62: ckm_gostr3410_512 ......................   Passed    1.79 sec
      Start 63: ckm_gostr3410_key_derive
63/87 Test #63: ckm_gostr3410_key_derive ...............   Passed    2.14 sec
      Start 64: ckm_gostr3410_12_256_key_derive
64/87 Test #64: ckm_gostr3410_12_256_key_derive .........   Passed    2.05 sec
      Start 65: ckm_gostr3410_12_512_key_derive
65/87 Test #65: ckm_gostr3410_12_512_key_derive .........   Passed    2.21 sec
      Start 66: ckm_gostr3410_2012_256_vko_256
```

```
66/87 Test #66: ckm_gostr3410_2012_256_vko_256 ..........   Passed    2.07 sec
      Start 67: ckm_gostr3410_2012_512_vko_256
67/87 Test #67: ckm_gostr3410_2012_512_vko_256 ..........   Passed    1.36 sec
      Start 68: ckm_gostr3410_2012_512_vko_512
68/87 Test #68: ckm_gostr3410_2012_512_vko_512 ..........   Passed    2.21 sec
      Start 69: ckm_gostr3410_key_wrap
69/87 Test #69: ckm_gostr3410_key_wrap ..................   Passed    3.38 sec
      Start 70: ckm_gostr3410_with_gostr3411
70/87 Test #70: ckm_gostr3410_with_gostr3411 ............   Passed    1.27 sec
      Start 71: ckm_gostr3410_with_gostr3411_12_256
71/87 Test #71: ckm_gostr3410_with_gostr3411_12_256 .....   Passed    1.41 sec
      Start 72: ckm_gostr3410_with_gostr3411_12_512
72/87 Test #72: ckm_gostr3410_with_gostr3411_12_512 .....   Passed    1.45 sec
      Start 73: cka_always_authenticate
73/87 Test #73: cka_always_authenticate .................   Passed    2.34 sec
      Start 74: keypair_import
74/87 Test #74: keypair_import ..........................   Passed    1.42 sec
      Start 75: ckm_tls_gost_prf
75/87 Test #75: ckm_tls_gost_prf ........................   Passed    1.05 sec
      Start 76: ckm_tls_gost_prf_2012
76/87 Test #76: ckm_tls_gost_prf_2012 ...................   Passed    1.04 sec
      Start 77: ckm_tls_gost_pre_master_key_gen
77/87 Test #77: ckm_tls_gost_pre_master_key_gen .........   Passed    0.87 sec
      Start 78: ckm_tls_gost_master_key_derive
78/87 Test #78: ckm_tls_gost_master_key_derive ..........   Passed    1.00 sec
      Start 79: ckm_tls_gost_key_and_mac_derive
79/87 Test #79: ckm_tls_gost_key_and_mac_derive .........   Passed    1.02 sec
      Start 80: ckm_tls12_master_key_derive
80/87 Test #80: ckm_tls12_master_key_derive .............   Passed    1.03 sec
      Start 81: ckm_tls12_key_and_mac_derive
81/87 Test #81: ckm_tls12_key_and_mac_derive ............   Passed    1.00 sec
      Start 82: ckm_tls_mac
82/87 Test #82: ckm_tls_mac .............................   Passed    1.08 sec
      Start 83: ckm_tls_kdf
83/87 Test #83: ckm_tls_kdf .............................   Passed    0.97 sec
      Start 84: ckm_tls_tree_gostr3411_2012_256
84/87 Test #84: ckm_tls_tree_gostr3411_2012_256 .........   Passed    0.92 sec
      Start 85: ckm_pkcs5_pbkd2
85/87 Test #85: ckm_pkcs5_pbkd2 .........................   Passed    1.05 sec
      Start 86: ckm_pba_gostr3411_with_gostr3411_hmac
86/87 Test #86: ckm_pba_gostr3411_with_gostr3411_hmac ...   Passed    0.99 sec
      Start 87: create_obj
87/87 Test #87: create_obj ..............................   Passed    7.67 sec
```

```
100% tests passed, 0 tests failed out of 87

Total Test time (real) = 114.13 sec
```

5 Ссылки

1. Официальный сайт ООО "ЛИССИ-Софт". - http://http://soft.lissi.ru//.
2. Официальный сайт Технического комитета по стандартизации (ТК 26) "Криптографическая защита информации". - https://www.tc26.ru.
3. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. – http://protect.gost.ru/document.aspx?control=7&id=139177.
4. ГОСТ Р 34.10-2001. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. – http://protect.gost.ru/document.aspx?control=7&id=131131.
5. ГОСТ Р 34.10-2012. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. – Москва, Стандартинформ, 2012.
6. ГОСТ Р 34.12-2015. Блочные шифры. – Москва, Стандартинформ, 2015.
7. ГОСТ Р 34.13-2015. Режимы блочных шифров. – Москва, Стандартинформ, 2015.
8. ГОСТ Р 34.11-94. Информационная технология. Криптографическая защита информации. Функция хеширования. – http://protect.gost.ru/document.aspx?control=7&id=134550.
9. ГОСТ Р 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хеширования. – Москва, Стандартинформ, 2012.
10. PKCS#11 v2.40: Cryptographic Token Interface Standard. - OASIS PKCS#11 TC. - https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=pkcs11.
11. Расширение PKCS#11 для использования российских криптографических алгоритмов. – Технический комитет по стандартизации (ТК 26) "Криптографическая защита информации". – Москва, ТК 26, 2008.
12. Расширение PKCS#11 для использования российских стандартов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 (готовится к публикации). – Технический комитет по стандартизации (ТК 26) "Криптографическая защита информации". – Москва, ТК 26, 2013.

13. Расширение PKCS#11 для использования российских стандартов ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 (готовится к публикации). – Технический комитет по стандартизации (ТК 26) "Криптографическая защита информации". – Москва, ТК 26, 2016.