Криптографическая диалоговая Lua-утилита для OpenSSL и PKCS#11

ООО "ЛИССИ-Софт"

30 апреля 2014 г.

Оглавление

1 Введение

Утилита lsp11ssl является диалоговым приложением, написанным на языке Lua [20]. Утилита позволяет выполнять криптографические операции PKCS#11 [17] и OpenSSL [16] в диалоговой форме, размещая исходные данные и результаты на токенах, во временных сессионных объектах, в файлах или в переменных Lua. Утилита позволяет легко добавлять новые функции или модифицировать существующие путем текстового редактирования Lua-скриптов программы. Никакой компиляции и сборки при этом не требуется.

Утилита поддерживает выполнение алгоритмов российской криптографии, определенных руководящими документами Технического комитета по стандартизации "Криптографическая защита информации" ТК 26 [2], включая алгоритмы ГОСТ Р34.10-2012 [5] и ГОСТ Р34.11-2012 [7].

2 Назначение

Утилита рассматривается в качестве приложения, демонстрирующего технологию создания легко модифицируемых кросс-платформенных графических программ, реализующих криптографические функции.

Утилита полезна при тестировании прикладных программ, использующих различные криптографические операции на уровне механизмов стандарта PKCS#11 и интерфейсов OpenSSL, потому что результаты выполнения этих операций можно сразу же увидеть визуально. Кроме того, утилита предоставляет удобный интерфейс для работы со списком объектов токена и их атрибутами.

Функциональность утилиты не претендует ни на исчерпывающую полноту охвата потребностей в предметной области, ни на совершенство реализации отдельных функций. Однако набор разнообразных функций в меню утилиты показывает, что сравнительно простым способом можно реализовать практически любой полезный инструментарий в диалоговой форме.

Демонстрируемый технологический подход может быть использован для создания вспомогательных программных инструментов администраторов безопасности информации, разработчиков и тестировщиков СКЗИ.

3 Особенности реализации

Для организации диалогов используются модули wxLua [18], являющиеся интерфейсными обертками вокруг библиотек wxWidgets [19]. Для доступа к библиотекам PKCS#11 используется интерфейсный модуль luacryptoki компании "ЛИССИ-Софт". Для выполнения функций OpenSSL из скриптов Lua используется интерфейсный модуль luaopenssl, также разработанный компанией "ЛИССИ-Софт"для поддержки российских криптографических алгоритмов. Предполагается, что пути к загружаемым Lua-модулям заданы в переменных среды `LUA_PATH` и `LUA_CPATH`, а сами модули установлены в системе соответствующим образом.

Модуль luaopenssl использовал в качестве прототипа аналогичный модуль luaopenssl [21], не поддерживающий российскую криптографию. Следует отметить, что в реализации модуля luaopenssl задействован универсальный механизм поддержки иерархии классов объектов, первоначально созданный в проекте LuaSocket [22]. Именно с помощью данного механизма в скриптах Lua единообразно обеспечивается обработка интерфейсных объектов OpenSSL типа `EVP_PKEY`, `X509`, `EVP_CIPHER`, `EVP_MD`, `PKCS7`, `PKCS12` и т.д.

Модуль luaopenssl использует динамические библиотеки LCSSL – модифицированной компанией "ЛИССИ-Софт"версии OpenSSL 1.0.0, поддерживающей алгоритмы российской криптографии. Поэтому для работы утилиты библиотеки LCSSL должны быть установлены в системе. Продукт LCSSL можно скачать с сайта ООО "ЛИССИ-Софт"[1].

Для использования функций PKCS#11 пользователь должен обеспечить утилите доступ к соответствующим динамическим библиотекам, реализующим данный интефейс. Компания "ЛИССИ-Софт"предлагает различные варианты таких библиотек [1]. Утилита может работать и с библиотеками других производителей.

Пользователь может изменить состав и структуру диалогов утилиты, а также порядок выполнения криптографических операций по своему усмотрению, модифицируя соответствующие скрипты. Основным скриптом программы, содержащим алгоритмы выполнения прикладных операций, является файл lcp11ssl.lua. Используются также скрипты start.lua, P11Ctx.lua, P11Names.lua, OpenSSLNames.lua, Factory.lua, TokenObject.lua, Template.lua, Attribute.lua, LuaData.lua, oids.lua, wxid.lua, hex.lua, csrdlg.lua, x509dlg.lua.

Проект утилиты, в принципе, кросс-платформенный. Однако, его разработка в макетном состоянии производится пока что только для Windows.

4 Запуск

Утилита может быть запущена с помощью простейшего интерпретатора:

```
wxLuaFreeze.exe start.lua
```

Заметим, что если wxLuaFreeze.exe собран статически, то динамическая библиотека wx.dll вообще не используется.

Средствами wxLua можно изготовить из стартового скрипта утилиты выполняемый файл. При этом, стартовый скрипт присоединяется в конец выполняемого файла wxLuaFreeze.exe и получает новое имя. Например, из папки scripts достаточно выполнить команду:

```
lua.exe wxluafreeze.lua wxluafreeze.exe start.lua lsp11ssl.exe
```

для генерации выполняемого файла lsp11ssl.exe.

Размер файла lsp11ssl.exe может быть уменьшен раза в три с помощью копрессионной утилиты upx:

```
upx lsp11ssl.exe
```

В результате запуска lsp11ssl.exe на экран выдается окно с главным меню (Рис. 4.1).

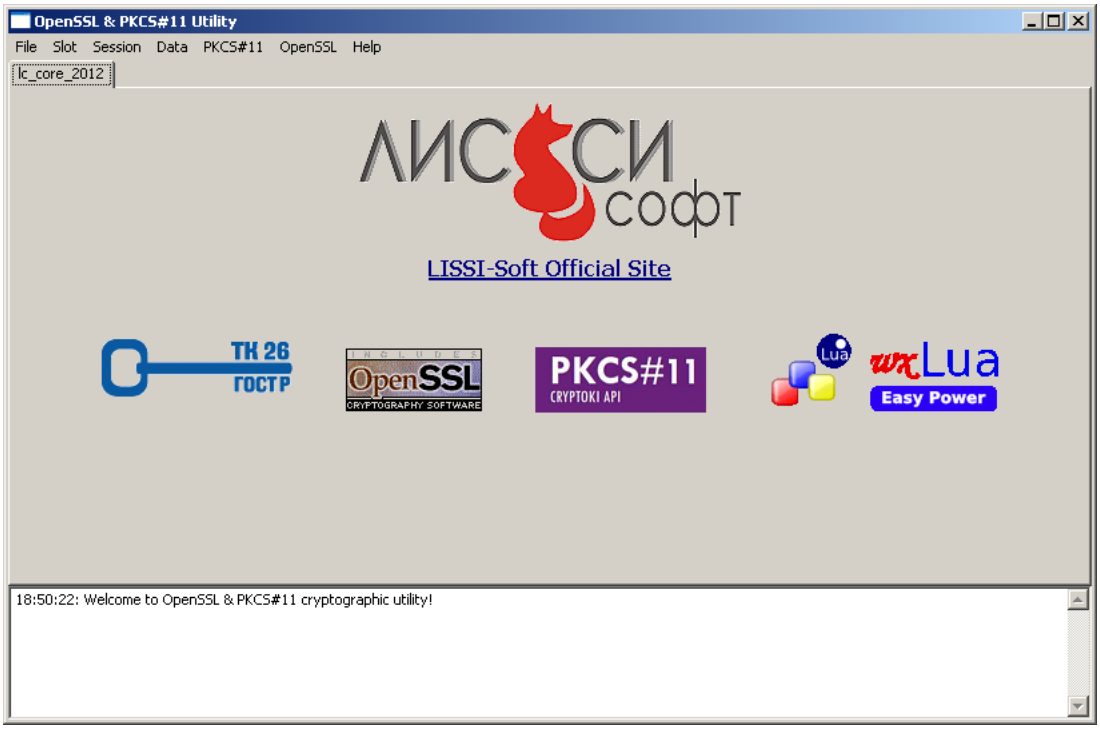

Рис. 4.1: Главное окно

В центральной части окна расположена начальная вкладка с логотипами и с контекстом OpenSSL, в котором используется ENGINE `lc_core_2012`. Данный контекст первоначально является текущим и всегда активным.

В нижней части находится окно диагностических сообщений утилиты.

Содержимое данного документа доступно для просмотра в утилите из пункта меню Help/Manual.

Из пункта меню Help/About выдается краткая справка об утилите (Рис. 4.2).

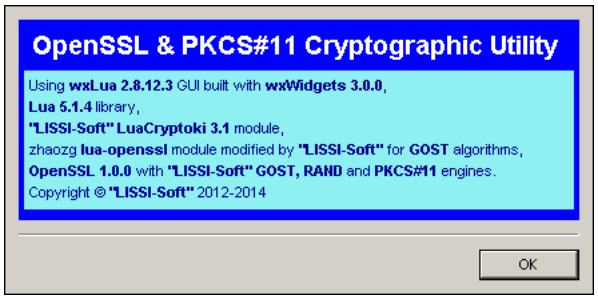

Рис. 4.2: Справка

5 Контексты PKCS#11 и OpenSSL

При запуске утилиты сразу открывается контекст OpenSSL. Данный контекст всегда активен и закрывается только при завершении работы утилиты.

Контекст PKCS#11 открывается из пункта Open меню File. При этом выбирается динамическая библиотека PKCS#11, реализующая данный интерфейс, на экране появляется дополнительная вкладка контекста данной библиотеки с именем библиотеки в заголовке вкладки. Этот контекст автоматически становится текущим (Рис. 5.1).

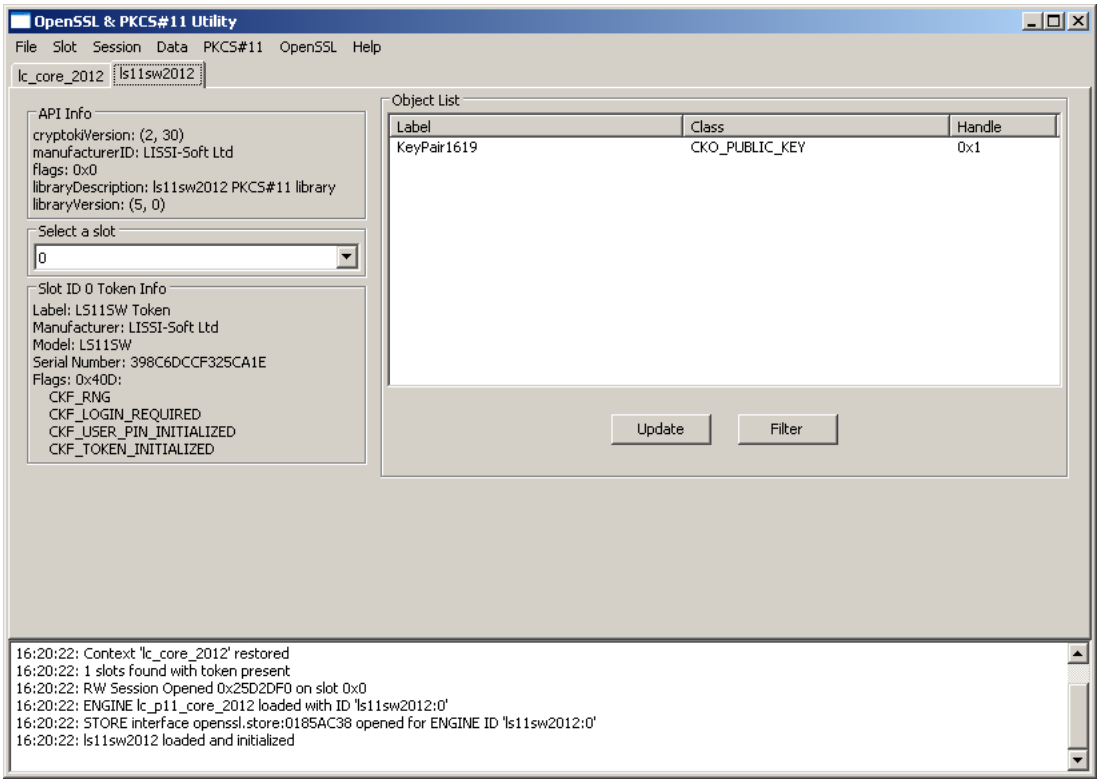

Рис. 5.1: Контекст PKCS#11

Здесь мы видим результат открытия контекста библиотеки программного токена ls11sw2012. При открытии контекста на слоте программного токена автоматически открывается сессия, но логин не выполняется, поэтому в окне объектов видны только публичные объекты токена.

Переключение между токенами, подключенными в контексте PKCS#11, производится с помощью комбобокса выбора слота на вкладке контекста.

Заметим, что хотя вкладка переключилась на контекст PKCS#11, контекст OpenSSL остается активным, поэтому функции OpenSSL также доступны. Более того, некоторые функции OpenSSL могут использовать объекты токена.

Утилита позволяет подключить одну или несколько библиотек PKCS#11 и переключать текущий контекст выполнения операций между разными токенами, не завершая открытую на слоте сессию, путем переключения библиотеки выбором соответствующей вкладки в главном окне или выбора другого слота на вкладке соответствующей текущей библиотеки. Выполнение прикладных функций всегда производится в текущем контексте, т.е. с одним токеном или вообще без токена в случае выбора контекста OpenSSL (Рис. 5.2).

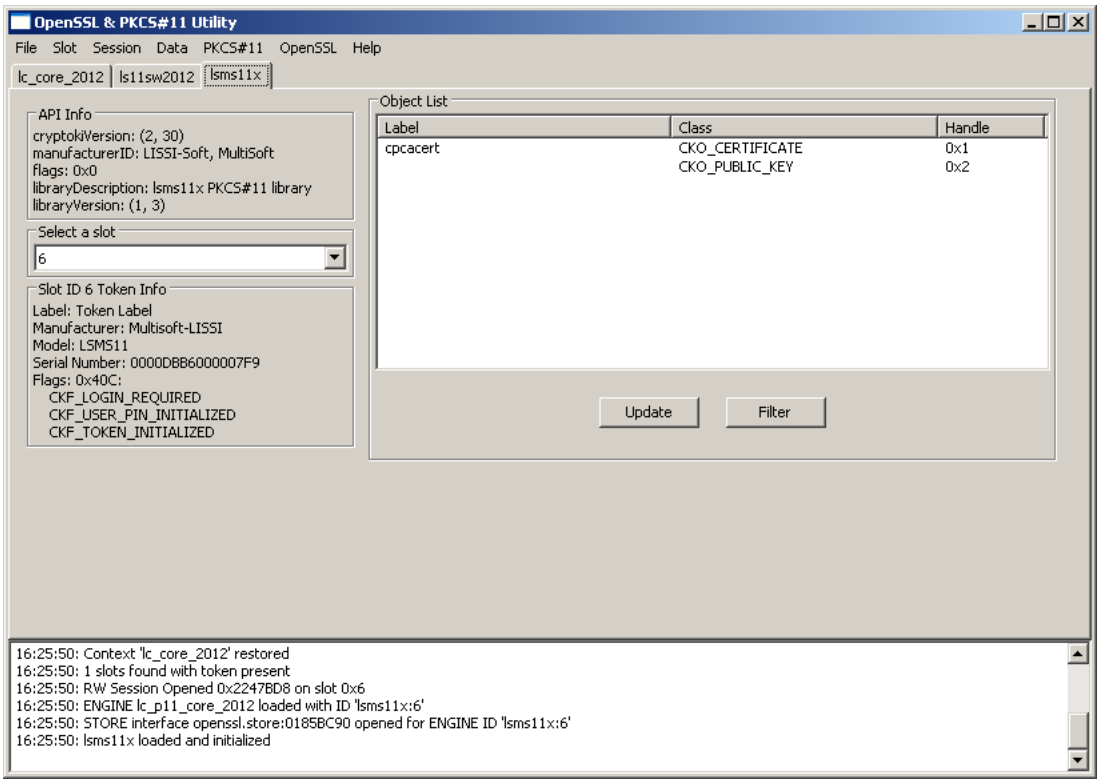

Рис. 5.2: Два контекста PKCS#11

Теперь дополнительно открыт контекст библиотеки lsms11x для доступа к токенам `MS_KEY K`. На текущей вкладке мы видим публичные объекты токена в слоте 6.

Утилита работает только с теми слотами, в которых она обнаружила токены при открытии библиотеки.

Если активен контекст PKCS#11 (открыта и выбрана соответствующая вкладка библиотеки PKCS#11 в главном окне), то доступны операции PKCS#11 и OpenSSL,

причем операции OpenSSL выполняются через ENGINE `lc_p11_core_2012`, связанный с соответствующим токеном. Особенностью `lc_p11_core_2012` является то, что он выполняет с помощью токена только операции с объектами ключевой пары, а все остальные операции (хеширование, симметричное шифрование и др.) поддерживаются в нем криптографической библиотекой LISSI-Crypto Core (lcc2012).

Утилита использует специально собранную версию OpenSSL 1.0.0, которая называется LCSSL. Для того, чтобы утилита могла загрузить ENGINE, путь к папке, где хранятся `lc_core_2012.dll`, `lc_p11_core_2012.dll` и `lc_rand.dll`, следует прописать в переменной среды `LCSSL_ENGINES`.

Заметим, что для многих операций с токеном требуется логин, поэтому нужно не забывать выполнять его в пункте Login меню Session. Если логин не произведен, то в списке объектов токена на странице контекста не будут представлены приватные объекты, а некоторые функции вообще нельзя будет выполнить. Добавим также, что для токенов различного типа могут действовать различные требования для авторизации пользователя при выполнении операций.

Если контекст PKCS#11 не активен (выбрана вкладка `lc_core_2012` в главном окне), то механизмы PKCS#11 не доступны, а криптографические операции выполняются только через OpenSSL и ENGINE `lc_core_2012`, работающий без токенов и поддерживающий все российские криптографические алгоритмы с помощью библиотеки LCC.

6 Меню Slot

Пункты данного меню позволяют получать различную информацию о слоте и токене, а также инициализировать токен.

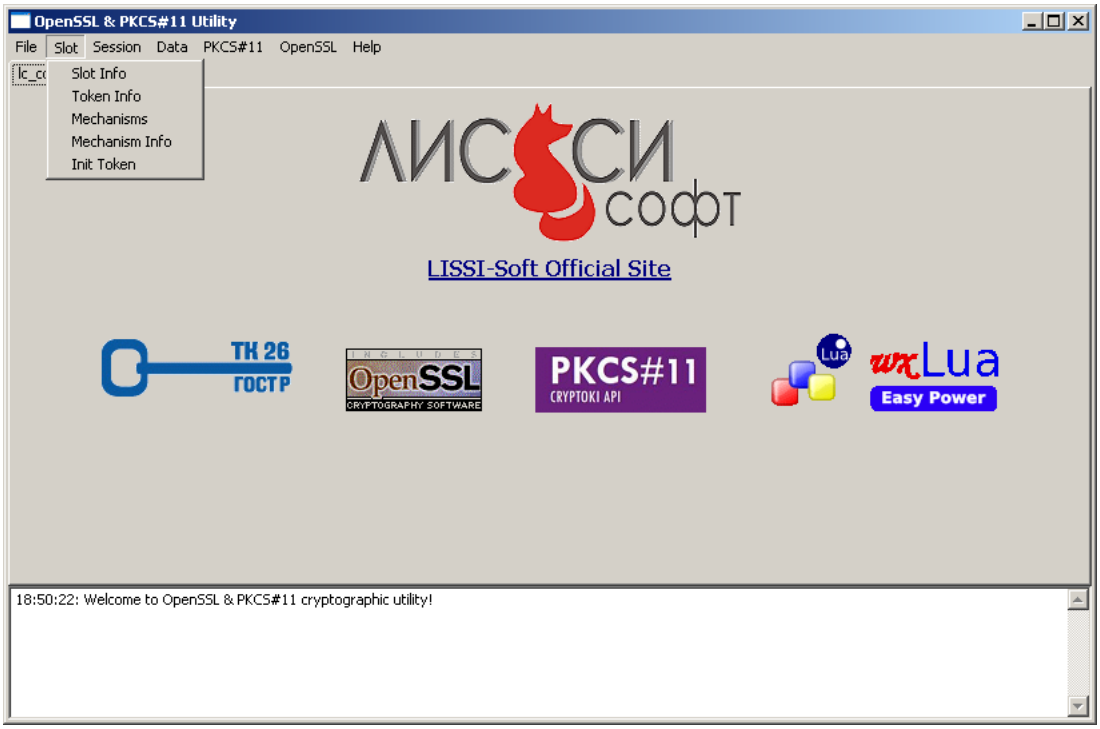

Рис. 6.1: Меню Slot

6.1 Информация о слоте

Рис. 6.2: Информация о слоте

6.2 Информация о токене

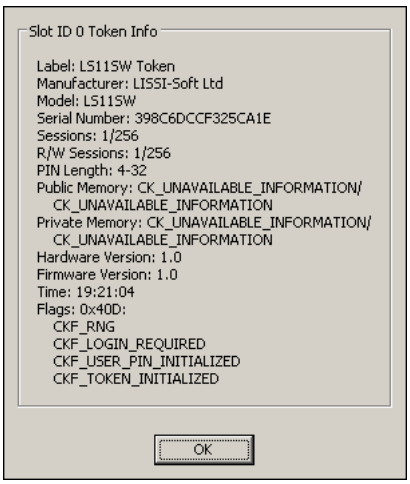

Рис. 6.3: Информация о токене

6.3 Список механизмов

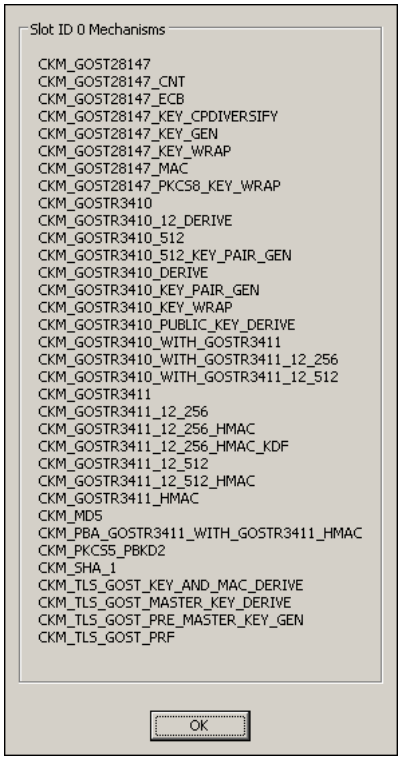

Рис. 6.4: Список механизмов

6.4 Информация о механизме

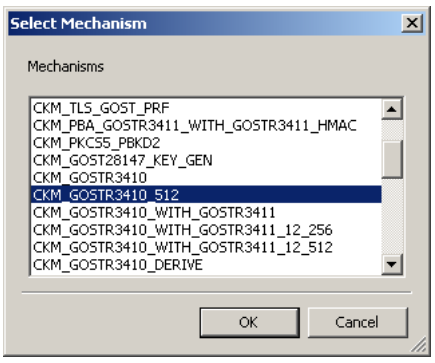

Рис. 6.5: Выбор механизма

Рис. 6.6: Информация о механизме

6.5 Инициализация токена

При выборе пункта меню Init Token выдается предупредительное сообщение (Рис. 6.7).

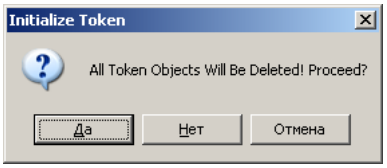

Рис. 6.7: Подтверждение или отмена инициализации токена

Если нажать OK, то после запроса SO PIN и метки токен будет инициализирован. При этом, все объекты на нем будут уничтожены.

7 Меню Session

При открытии контекста PKCS#11 на каждом слоте с токеном открывается умалчиваемая сессия пользователя для чтения и записи. Из меню Session можно производить операции с сессиями (Рис. 7.1).

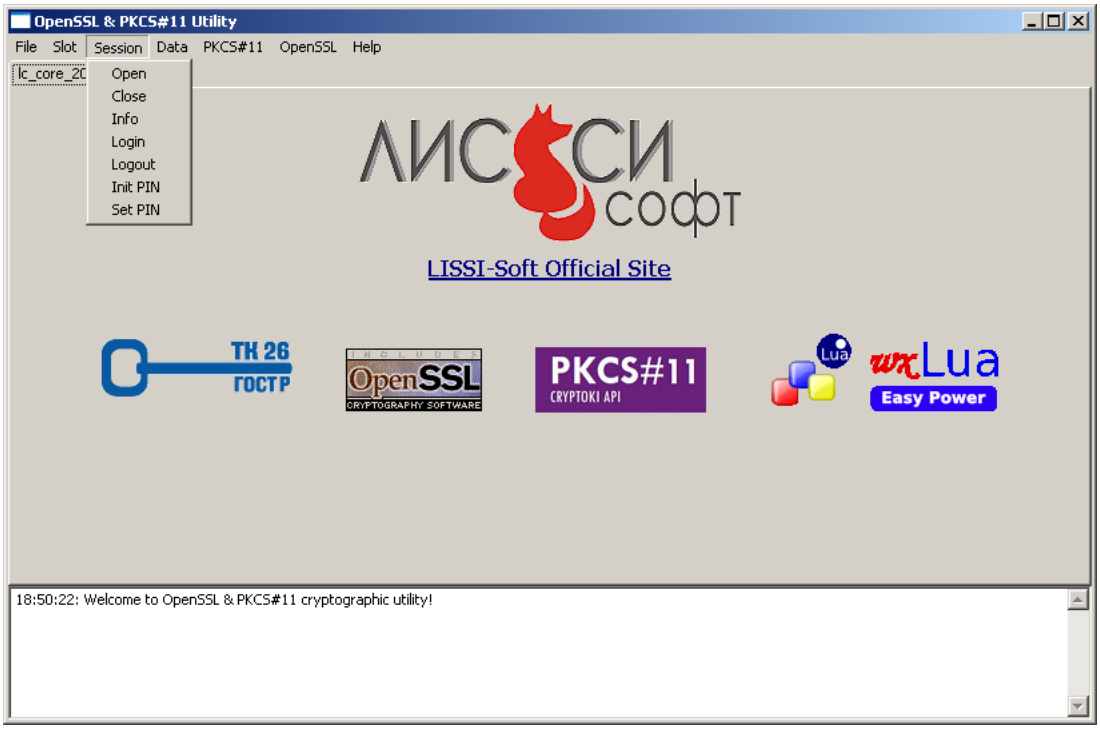

Рис. 7.1: Меню Session

7.1 Информация о сессии

Информация о текущей сессии выдается при выборе пункта Info (Рис. 7.2).

Рис. 7.2: Информация о сессии

7.2 Открытие и закрытие сессии

Текущая сессия может быть закрыта и вместо нее можно открыть другую сессию. Это может понадобиться, например, если требуется поработать с токеном в режиме только для чтения.

7.3 Логин и логаут

При открытии контекста PKCS#11 логин не выполняется, поэтому в списке объектов видны только публичные объекты. При выборе пункта Login запрашивается тип пользователя – User или SO, после чего производится запрос соответствующего PIN. Если успешно выполнен логин User, то в списке объектов появляются и приватные объекты. Если выполнен логин SO, то разрешается выполнять функции SO, причем приватные объекты отображаться не будут.

При выборе пункта Logout сессия возвращается в неавторизованное состояние. Приватные объекты при этом не видны.

7.4 Инициализация и установка PIN

После инициализации токена пользовательский PIN на нем не установлен. Начальная установка пользовательского PIN выполняется администратором безопасности в SO-логине выбором пункта Init PIN. Текже в SO-логине может быть изменено значение SO PIN выбором пункта Set PIN.

Новое значение PIN пользователь устанавливает самостоятельно в User-логине выбором пункта Set PIN.

8 Обработка данных

В различных функциях утилиты имеется возможность загрузки исходных данных для выполнения прикладных операций из файлов или из переменных Lua. Например, из пункта меню Data/Data можно вызвать диалог загрузки данных. Результирующие данные могут быть сохранены в файле или во временной переменной Lua (Рис. 8.1).

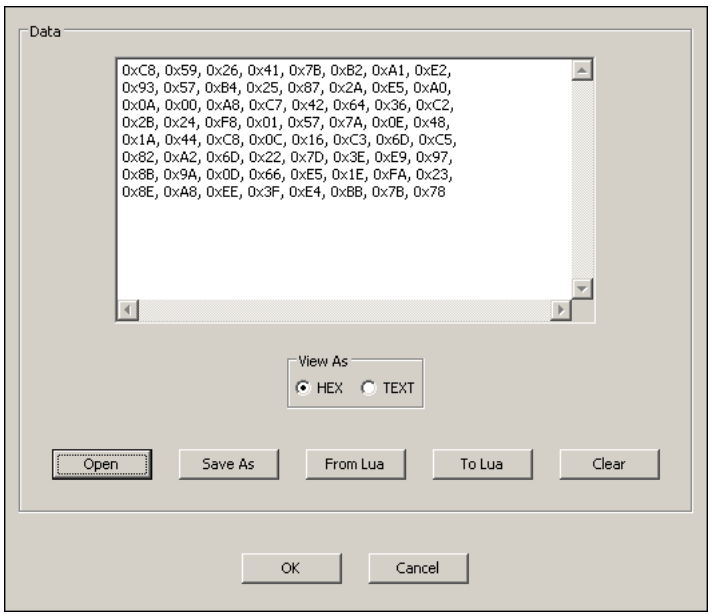

Рис. 8.1: Диалог обработки данных

После загрузки данные отображаются в окне в шестнадцатеричном виде. Для отображения их в текстовой форме можно использовать переключатель HEX-TEXT.

Данные могут быть сгенерированы и случайным образом средствами OpenSSL из пукта меню Data/Random, причем в пункте Data/RAND Engine имеется возможность задать умалчиваемый ENGINE для интерфейса RAND, например - `lc_rand`.

При выборе пункта Data/Random Seed выдается диалог для генерации уникального начального значения датчика случайных чисел с использованием перемещения мыши по полю диалога (Рис. 8.2).

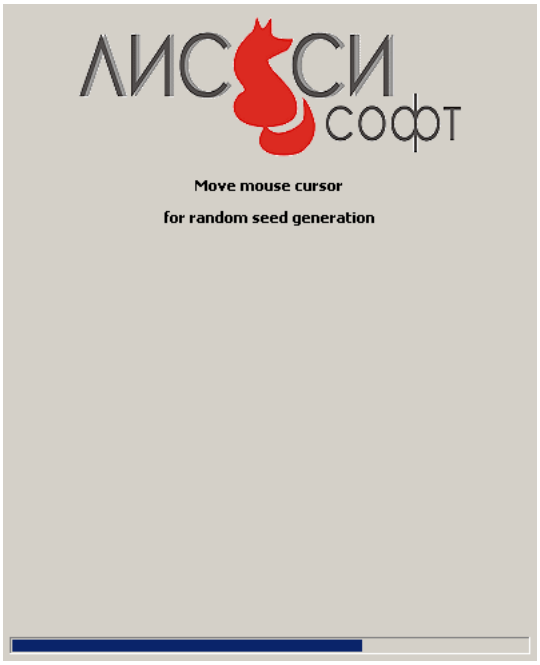

Рис. 8.2: Генерация начального значения ДСЧ

С помошью пункта меню Data/ASN.1 Parse можно визуализировать содержимое файла в формате DER в виде древовидной структуры (Рис. 8.3).

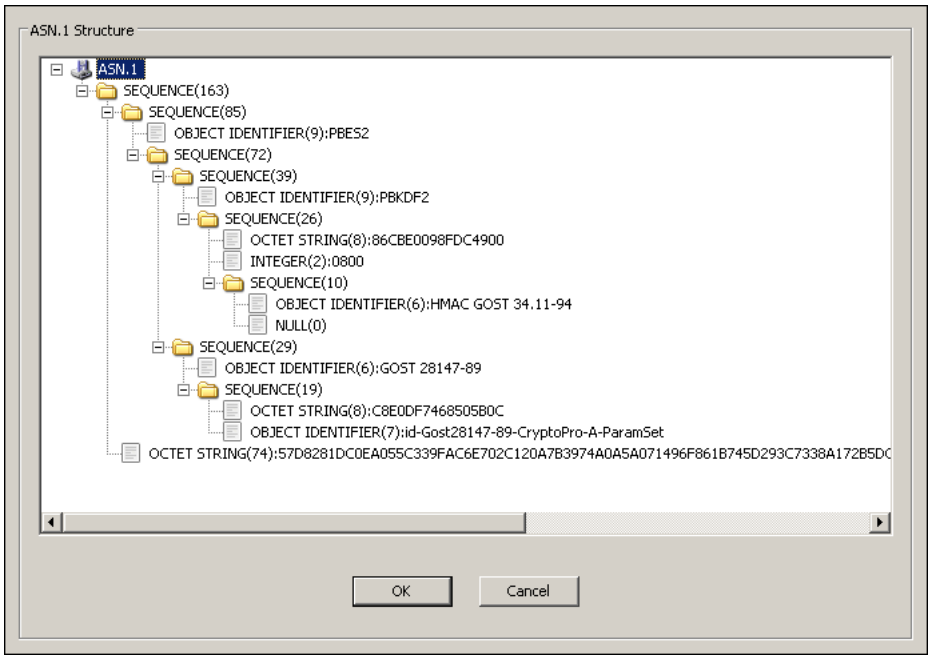

Рис. 8.3: ASN.1-структура

9 Меню PKCS#11

Пункты данного меню выполняются только при открытой сессии в контексте PKCS#11. Более того, некоторые пункты требуют еще и авторизации пользователя в логине.

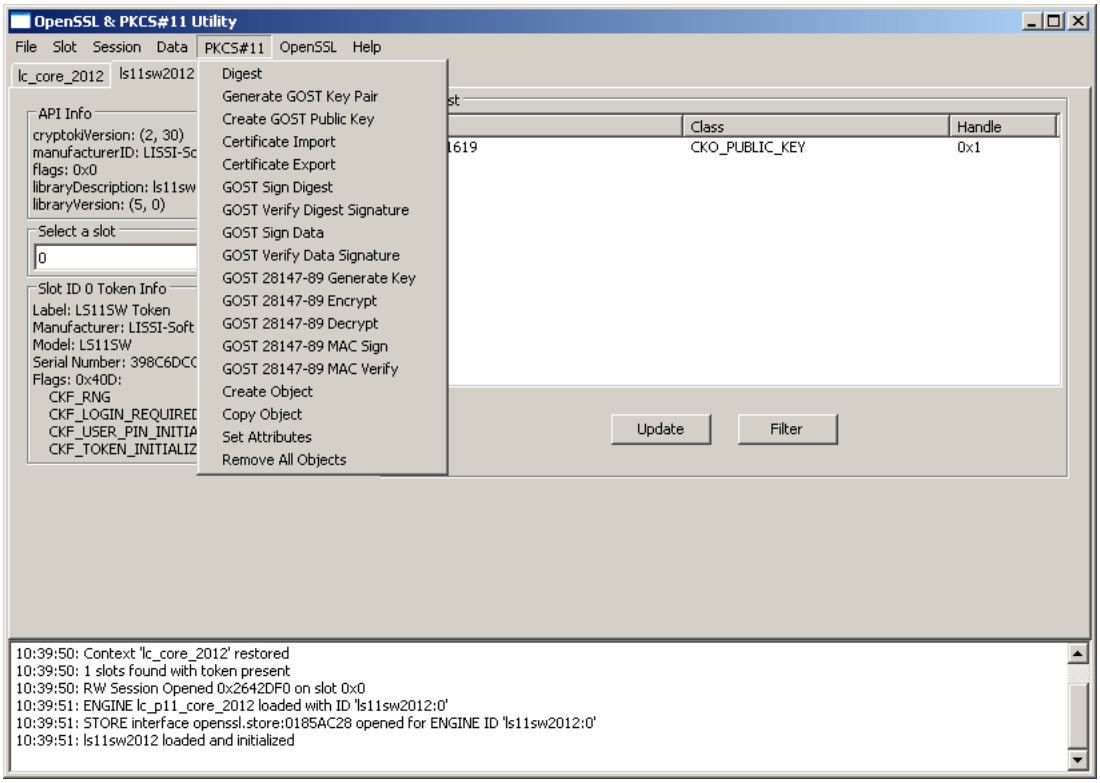

Рис. 9.1: Меню PKCS#11

9.1 Работа с объектами токена

Все объекты текущего слота отображаются в списке объектов на вкладке контекста. Заметим, что отображаются и объекты токена, и временные сессионные объекты. Если не выполнен логин, то отображаются лишь публичные объекты. Если текущую сессию закрыть, то объекты вообще отображаться не будут.

Состав объектов PKCS#11 и значения их атрибутов могут быть изменены пользователем в соответствующих диалогах, если такая операция допустима по стандарту.

Удаление объекта производится путем его выбора левой кнопкой мыши из списка и последующего нажатия клавиши Delete. Выбрав несколько объектов при нажатой клавише Ctrl, можно удалить все выбранные объекты.

Вообще говоря, объекты токена могут быть изменены другим приложением во время работы утилиты или при доступе к одному и тому же токену из разных контекстов. В такой ситуации для обновления отображаемого в текущем контексте состояния объектов можно нажать кнопку Update под списком объектов.

Утилита позволяет редактировать шаблоны атрибутов, используемые в объектах. Диалог атрибутов открывается при двойном щелчке мыши на объекте токена из списка (Рис. 9.2).

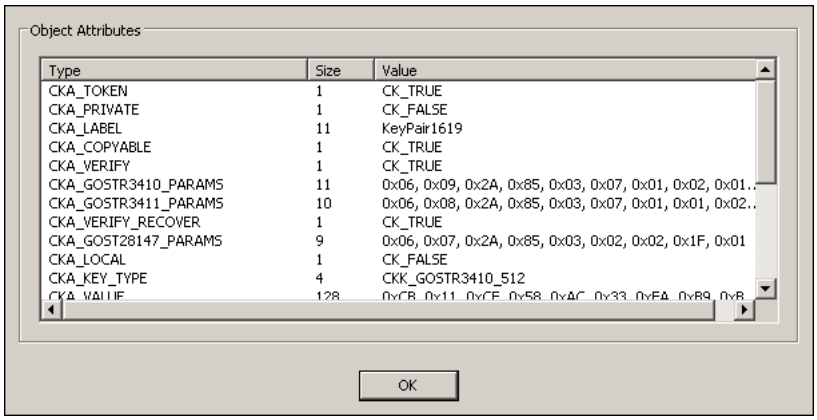

Рис. 9.2: Шаблон атрибутов объекта

Двойной щелчок мыши на атрибуте объекта открывает диалог редактирования значения атрибута (Рис. 9.3).

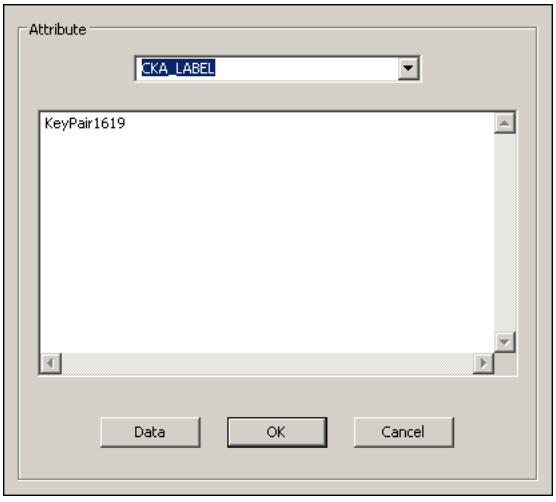

Рис. 9.3: Редактирование значения атрибута

9.2 Генерация дайджеста

При выборе данного пункта меню сначала выдается диалог, запрашивающий механизм дайджеста, затем выдается диалог ввода хэшируемых данных, после чего в диалоге результата предлагается сохранить значение дайджеста.

9.3 Генерация ключевой пары

При выборе данного пункта меню сначала выдается диалог, запрашивающий механизм генерации ключевой пары, затем поочередно выдаются диалоги редактирования исходных шаблонов открытого и закрытого ключей, после чего в дополнительном диалоге предлагается для связи открытого ключа с закрытым установить в закрытом ключе значение атрибута `CKA_ID`, равное дайджесту SHA-1 от значения открытого ключа.

9.4 Создание открытого ключа

Иногда значение открытого ключа поступает в виде последовательности байтов, а функции PKCS#11 работают только с хэндлами объектов. Для создания объекта в данном пункте меню предлагается исходный шаблон открытого ключа с возможностью его редактирования. После нажатия кнопки OK создается соответствующий объект PKCS#11 и появляется в списке объектов слота.

9.5 Импорт и экспорт сертификата

Сертификат X.509 может быть загружен в качестве значения объекта PKCS#11 класса `CKO_CERTIFICATE` и типа `CKC_X_509`.

9.6 Генерация и проверка ЭЦП

Для генерации и проверки ЭЦП имеются механизмы, работающие с готовым дайджестом подписываемых данных, и комбинированные механизмы, сами вырабатывающие дайджест по исходным данным. Для разных механизмов и функций используются соответствующие пункты меню.

9.7 Генерация ключа ГОСТ 28147-89

При выборе данного пункта меню сначала выдается диалог, запрашивающий OID используемого набора параметров. Затем выдается для редактирования исходный шаблон атрибутов ключа, а после нажатия кнопки OK новый ключ появляется в списке объектов.

9.8 Симметричное шифрование

Для зашифрования предлагается выбрать исходный открытый текст, объект ключа шифрования и указать значение IV. Затем зашифрованный текст предлагается сохранить в файле или в переменной Lua.

При расшифровании задается зашифрованный текст, объект ключа шифрования и значение IV. Затем расшифрованный текст предлагается сохранить в файле или в переменной Lua.

9.9 Имитовставка

Для генерации имитовставки предлагается выбрать исходные данные, объект ключа шифрования и указать значение IV. Затем значение имитовставки предлагается сохранить в файле или в переменной Lua.

Для проверки имитовставки предлагается выбрать исходные данные, объект ключа, IV и значение имитовставки. Результат проверки отображается в окне сообщений.

9.10 Создание объекта

При выборе данного пункта меню имеется возможность создать любой объект PKCS#11, вручную конструируя его шаблон.

9.11 Копирование объекта

При выборе данного пункта меню имеется возможность создать копию существующего объекта PKCS#11, изменяя некоторые атрибуты в его шаблоне, если это допускается стандартом.

9.12 Изменение атрибутов объекта

При выборе данного пункта меню имеется возможность изменить некоторые атрибуты в шаблоне существующего объекта, если это допускается стандартом.

9.13 Удаление всех объектов

При выборе данного пункта меню удаляются все объекты текущего слота, включая и объекты токена, и сессионные объекты.

10 Меню OpenSSL

Выполнение функций OpenSSL возможно в контексте OpenSSL или в контексте PKCS#11. В контексте OpenSSL объекты PKCS#11 не доступны.

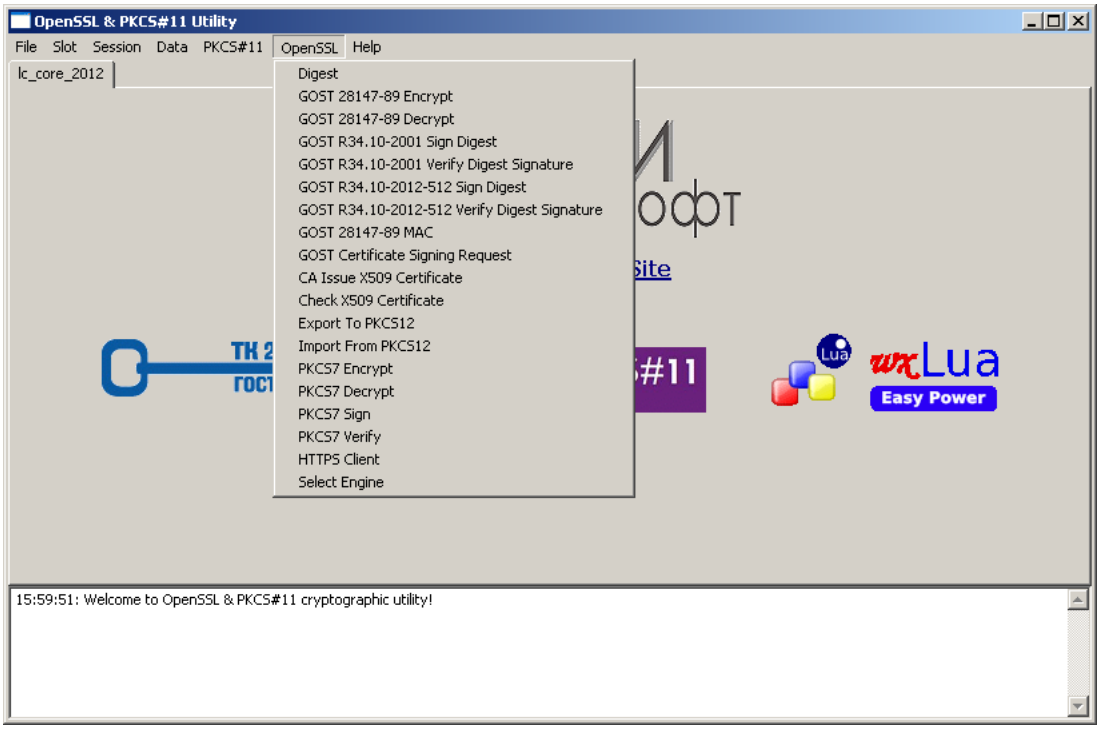

Рис. 10.1: Меню OpenSSL

В контексте PKCS#11 возможно выполнение функций OpenSSL с использованием объектов токена. Доступ к объектам токена из OpenSSL обеспечивается с помощью ENGINE `lc_p11_core_2012`, который, в свою очередь, задействует библиотеку PKCS#11, открытую в текущем контексте. Если пользователь не указал объект токена, то будет предложено ввести его из файла средствами OpenSSL без использования интерфейса PKCS#11.

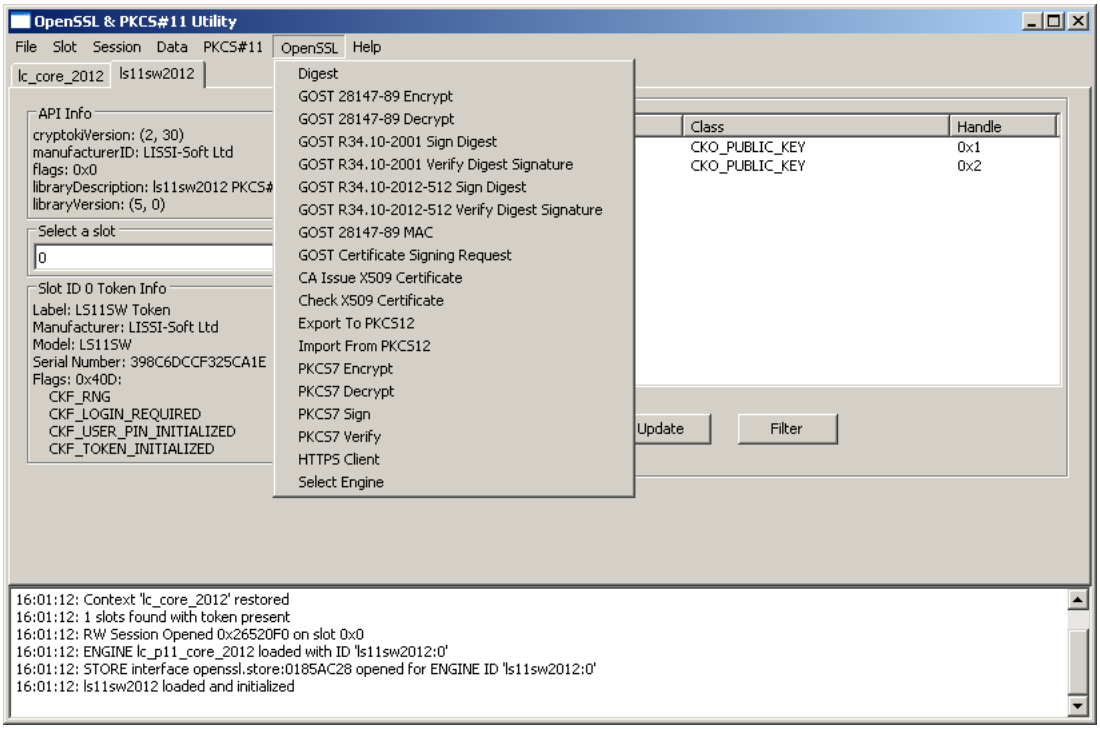

Рис. 10.2: Меню OpenSSL в контексте PKCS#11

10.1 Генерация дайджеста

При выборе данного пункта меню сначала выдается диалог, запрашивающий тип алгоритма дайджеста, затем выдается диалог ввода хэшируемых данных, после чего в диалоге результата предлагается сохранить значение дайджеста.

10.2 Генерация и проверка ЭЦП

Для генерации и проверки ЭЦП имеются функции, работающие с готовым дайджестом подписываемых данных, и комбинированные функции, сами вырабатывающие дайджест по исходным данным. Для разных функций используются соответствующие пункты меню.

10.3 Симметричное шифрование

Для зашифрования предлагается выбрать исходный открытый текст, значение ключа шифрования и указать значение IV. Затем зашифрованный текст предлагается сохранить в файле или в переменной Lua.

При расшифровании задается зашифрованный текст, значение ключа шифрования и значение IV. Затем расшифрованный текст предлагается сохранить в файле

или в переменной Lua.

10.4 Имитовставка

Для генерации имитовставки предлагается выбрать исходные данные и значение ключа шифрования. Затем значение имитовставки предлагается сохранить в файле или в переменной Lua. В реализации данного пункта IV не задается.

10.5 Запрос на сертификат

При выборе данного пункта генерируется новая ключевая пара и формируется запрос на сертификат в формате PKCS#10.

10.6 Выпуск сертификата

В данном пункте имитируется выпуск сертификата по запросу с подписью сертификата заданным пользователем закрытым ключом УЦ.

10.7 Экспорт в PKCS#12 и импорт из него

В соответствующих пунктах меню производится экспорт закрытого ключа и сертификата в транспортный контейнер PKCS#12 и импорт этих объектов из контейнера. Если открыт контекст PKCS#11, то возможно использование объектов токена, если библиотека и объекты допускают экспорт и импорт.

10.8 PKCS#7 – шифрование

В соответствующих пунктах меню производится зашифрование и расшифрование данных в структуре PKCS#7 с использованием ключей отправителя и получателя.

10.9 PKCS#7 – подпись и проверка

В соответствующих пунктах меню производится подпись и проверка подписи в структуре PKCS#7 с использованием ключей отправителя и получателя.

10.10 Клиент HTTPS

При выборе данного пункта меню сначала предлагается указать адрес сервера и номер порта, затем предлагается указать доверенный сертификат сервера, а также выбрать сертификат и закрытый ключ клиента. После этого устанавливается

защищенное соединение и производится запрос страницы сервера. Полученное содержимое страницы отображается в окне (Рис. 10.3).

Рис. 10.3: Страница, полученная с сервера HTTPS

10.11 Загрузка дополнительного ENGINE

По умолчанию, утилита загружает и использует в функциях OpenSSL криптографический ENGINE `lc_core_2012`. Если нужно дополнительно загрузить другой ENGINE (например, штатный ENGINE OpenSSL gost) и сделать его умалчиваемым, то его идентификатор запрашивается при выборе пункта меню OpenSSL/Load Engine. При этом, ENGINE `lc_core_2012` тоже остается загруженным и поддерживает те функции, которые не поддерживаются умалчиваемым ENGINE.

Заметим, что библиотека дополнительного ENGINE должна быть размещена там же, где и библиотека `lc_core_2012` – в папке lib/engines.

Данная функция не работает при открытом контексте PKCS#11.

11 Программирование

Преимуществом программирования на языке Lua является легкость внесения изменений в программу и возможность сразу получить результаты этих изменений, не производя компиляции и сборки программы.

Для изменения функциональности утилиты нужно уметь программировать на языке Lua с использованием интерфейсов wxLua к оконным средствам wxWidgets. Кроме того, для доступа к функциям PKCS#11 и OpenSSL нужно ознакомиться с интерфейсами luacryptoki и luaopenssl. Примерами использования этих интерфейсов являются Lua-скрипты утилиты. Основным скриптом утилиты является файл lsp11ssl.lua.

11.1 Загрузка модулей

Загрузка различных модулей и организация главного окна производятся в начале работы утилиты в скрипте lsp11ssl.lua.

```
-- Copyright ООО "ЛИССИ-Софт", 2012-2014

-- Это основной скрипт утилиты, содержащий алгоритмы выполнения
-- различных прикладных функций OpenSSL и PKCS#11.

-- Предполагается, что пути к модулям Lua заданы регулярными
-- выражениями в переменных среды LUA_PATH и LUA_CPATH.
-- Путь к OpenSSL engines (lc_core_2012, lc_p11_core_2012,
   lc_rand)
-- должен быть задан в переменной среды LCSSL_ENGINES.
-- Пути к остальным динамически загружаемым библиотекам
-- должны быть заданы в переменных среды PATH (для Windows)
-- или LD_LIBRARY_PATH (для Linux).

local OS = os.getenv("OS")
Windows = false
if OS and string.match(OS, "Windows") then
    Windows = true
end

-- Загрузка интерфейсных модулей ----------------------
-- Модуль wxLua, использующий Lua 5.1 и wxWidgets 2.9.5
```

```
require("wx")
-- Модуль bit включен в wxLua,
-- но мы будем использовать не bit, а bit32
-- для будущей совместимости с Lua 5.2.
bit32 = bit
-- Lua-интерфейс к функциям OpenSSL
local openssl = require("luaopenssl")
-- Lua-интерфейс к функциям PKCS#11
funcs = require("luacryptoki")
-- Lua-интерфейс к контексту генерации начального значения ДСЧ
local biorng = require("luabiorng")

-- Lua-интерфейс к библиотеке iconv используется для
-- преобразования текстов из UTF-8 и UTF-16
-- в наружную кодироку, заданную в переменной
-- visual_encoding.
local iconv = require("luaiconv")
-- Кодировка отображаемых данных зависит от типа сборки wxLua.
-- Поскольку по умолчанию wxLua теперь собирается для UTF-8, то
-- utf8tovisual и visualtoutf8 просто копируют данные
-- без преобразования.
-- Если бы тип сборки wxLua был ANSI, то нужно было бы
-- задавать для Windows:
-- visual_encoding = "cp1251".
visual_encoding = "utf-8"
-- to <= from
utf8tovisual = iconv.new(visual_encoding, "utf-8")
visualtoutf8 = iconv.new("utf-8", visual_encoding)
utf16tovisual = iconv.new(visual_encoding, "utf-16")
-- Эти перекодировщики могут понадобиться
-- для перекодировки текстовых данных:
utf8tocp1251 = iconv.new("cp1251", "utf-8")
cp1251toutf8 = iconv.new("utf-8", "cp1251")
utf16toutf8 = iconv.new("utf-8", "utf-16")
utf8toutf16 = iconv.new("utf-16", "utf8")
------------------------------------------------------------

-- Теперь остальные Lua-скрипты данной утилиты --------------
-- Модуль (класс) контекстов P11
P11Ctx = require("P11Ctx")
-- Набор констант и функций, предназначенный
-- для работы с данными P11 в текстовой форме.
require("P11Names")
-- Аналогичный набор для OpenSSL
```

```
require("OpenSSLNames")
-- OIDы КриптоПро и ТК 26
require("oids")
-- Идентификаторы виджетов
require("wxid")
-- Преобразователи строки байтов в шестнадцатеричный вид и обратно
require("hex")
-- Модуль объектов данных
LuaData = require("LuaData")
-- Модуль объектов токена
TokenObject = require("TokenObject")
-- Модуль шаблонов
Template = require("Template")
-- Модуль диалога CSR
csrdlg = require("csrdlg")
-- Модуль диалога X509
x509dlg = require("x509dlg")
-- Утилиты ввода-вывода
require("util")

-- Логотипы для умалчиваемой страницы в формате Lua-XPM -------
-- Логотип ЛИССИ-Софт
require("lissi_soft_color")
-- Логотип ТК 26
require("tc26")
-- Логотип OpenSSL
require("openssl_logo")
-- Логотип Cryptoki
require("Cryptoki")
-- Логотип wxLua
require("wxluasmall")

-- Используем русский язык в стандартных диалогах (если надо)
--local loc = wx.wxLocale(wx.wxLANGUAGE_RUSSIAN)

-- Главное окно с главным меню, страницами контекстов
-- и окном сообщений
frame = wx.wxFrame(wx.NULL, wx.wxID_ANY,
    "OpenSSL & PKCS#11 Utility",
    wx.wxDefaultPosition, wx.wxSize(820, 540))
local panel_frame = wx.wxPanel(frame, wx.wxID_ANY)
local sizer_panel = wx.wxBoxSizer(wx.wxVERTICAL)
panel_frame:SetSizer(sizer_panel)
```

```
-- Книга со страницами контекстов
local book = wx.wxNotebook(panel_frame, wx.wxID_ANY)
-- Указываем модулю контекстов книгу,
-- в которой размещаются страницы контекстов.
P11Ctx.SetBook(book)

-- Умалчиваемая страница с контекстом OpenSSL + lc_core_2012.
-- Функции PKCS#11 из нее не доступны, зато на ней
-- отображаются логотипы разработчика и используемых систем,
-- а также ссылка на сайт разработчика.
local panel_lissi = wx.wxPanel(book, wx.wxID_ANY)
local sizer_lissi = wx.wxBoxSizer(wx.wxVERTICAL)
panel_lissi:SetSizer(sizer_lissi)
-- Логотип ЛИССИ-Софт
local bitmap = wx.wxBitmap(lissi_soft_color)
bitmap:SetMask(wx.wxMask(bitmap, wx.wxWHITE))
local static_bitmap =
    wx.wxStaticBitmap(panel_lissi, wx.wxID_ANY, bitmap)
bitmap:delete()
sizer_lissi:Add(static_bitmap, 0, wx.wxCENTRE + wx.wxALL, 10)
-- Гиперссылка на сайт ЛИССИ-Софт
local ref = wx.wxHyperlinkCtrl(panel_lissi, wx.wxID_ANY,
    "LISSI-Soft Official Site",
    "http://soft.lissi.ru")
local font = ref:GetFont()
font:SetWeight(wx.wxFONTWEIGHT_BOLD)
font:SetPointSize(wx.wxNORMAL_FONT:GetPointSize() * 1.5)
ref:SetFont(font)
sizer_lissi:Add(ref, 0, wx.wxALIGN_CENTER, 10)

local sizer_logo_1 = wx.wxBoxSizer(wx.wxHORIZONTAL)

-- Логотип ТК 26
local bitmap_tc26 = wx.wxBitmap(tc26_logo_xpm)
bitmap_tc26:SetMask(wx.wxMask(bitmap_tc26, wx.wxWHITE))
local static_bitmap_tc26 =
    wx.wxStaticBitmap(panel_lissi, wx.wxID_ANY, bitmap_tc26)
bitmap_tc26:delete()
sizer_logo_1:Add(static_bitmap_tc26, 0, wx.wxCENTRE + wx.wxALL,
   20)

-- Логотип OpenSSL
local bitmap_openssl = wx.wxBitmap(openssl_logo_xpm)
```

```
bitmap_openssl:SetMask(wx.wxMask(bitmap_openssl, wx.wxWHITE))
local static_bitmap_openssl =
    wx.wxStaticBitmap(panel_lissi, wx.wxID_ANY, bitmap_openssl)
bitmap_openssl:delete()
sizer_logo_1:Add(static_bitmap_openssl, 0, wx.wxCENTRE +
   wx.wxALL, 20)

-- Логотип Cryptoki
local bitmap_cryptoki = wx.wxBitmap(cryptoki_logo_xpm)
bitmap_cryptoki:SetMask(wx.wxMask(bitmap_cryptoki, wx.wxWHITE))
local static_bitmap_cryptoki =
    wx.wxStaticBitmap(panel_lissi, wx.wxID_ANY, bitmap_cryptoki)
bitmap_cryptoki:delete()
sizer_logo_1:Add(static_bitmap_cryptoki, 0, wx.wxCENTRE +
   wx.wxALL, 20)

-- Логотип wxLua
local bitmap_wxlua = wx.wxBitmap(wxLuaSmall_xpm)
bitmap_wxlua:SetMask(wx.wxMask(bitmap_wxlua, wx.wxWHITE))
local static_bitmap_wxlua =
    wx.wxStaticBitmap(panel_lissi, wx.wxID_ANY, bitmap_wxlua)
bitmap_wxlua:delete()
sizer_logo_1:Add(static_bitmap_wxlua, 0, wx.wxCENTRE + wx.wxALL,
    20)

sizer_lissi:Add(sizer_logo_1, 0, wx.wxCENTRE + wx.wxALL, 20)

book:AddPage(panel_lissi, "lc_core_2012", true)
sizer_panel:Add(book, 1, wx.wxALIGN_CENTER + wx.wxGROW, 10)

-- Окно сообщений
log_ctrl = wx.wxTextCtrl(panel_frame, wx.wxID_ANY, "",
    wx.wxDefaultPosition, wx.wxSize(200, 100),
    wx.wxTE_MULTILINE + wx.wxTE_READONLY)
sizer_panel:Add(log_ctrl, 0, wx.wxGROW, 10)
-- Назначение вывода сообщений в данное окно
local logTargetOld =
    wx.wxLog.SetActiveTarget( wx.wxLogTextCtrl(log_ctrl) )

-- Далее идет организация меню, описанная ниже
-- ...

-- Выдача диалога
frame:Show(true)
```

```
-- Выдача стартового сообщения
wx.wxLogMessage("Welcome to OpenSSL & PKCS#11 cryptographic
   utility!")

-- Главный цикл обработки сообщений wxLua не запустится здесь,
-- если он уже запущен в вызывающей программе C++.
wx.wxGetApp():MainLoop()
```

11.2 Организация меню

Каждому пункту меню назначается уникальный целочисленный идентификатор. Список таких идентификаторов содержится в скрипте wxid.lua. Для некоторых стандартных пунктов типа Help/About можно задействовать идентификаторы, определенные в wxLua.

```
-- ...
-- Пункты меню
-- File menu
-- Функции открытия/закрытия контекстов PKCS#11 и выхода
local fileMenu = wx.wxMenu()
fileMenu:Append(ID_Open,
    "Open", "Open PKCS#11 Library")
fileMenu:Append(ID_Close,
    "Close", "Close PKCS#11 Library")
fileMenu:Append(ID_CloseAll,
    "CloseAll", "Close All PKCS#11 Libraries");
fileMenu:Append(wx.wxID_EXIT,
    "Exit", "Quit This Program")
-- Slot menu
-- Функции слотов и токенов
local slotsMenu = wx.wxMenu()
slotsMenu:Append(ID_SlotInfo,
    "Slot Info", "Get Slot Info")
slotsMenu:Append(ID_TokenInfo,
    "Token Info", "Get Token Info")
slotsMenu:Append(ID_Mechanisms,
    "Mechanisms", "Get Mechanisms")
slotsMenu:Append(ID_MechanismInfo,
    "Mechanism Info", "Get Mechanism Info")
slotsMenu:Append(ID_InitToken,
    "Init Token", "Initialize Token")
-- Session menu
-- Функции сессий, PIN и логинов
```

```
local sessionMenu = wx.wxMenu()
sessionMenu:Append(ID_OpenSession,
    "Open", "Open Session")
sessionMenu:Append(ID_CloseSession,
    "Close", "Close Session")
sessionMenu:Append(ID_SessionInfo,
    "Info", "Get Session Info")
sessionMenu:Append(ID_Login,
    "Login", "Login")
sessionMenu:Append(ID_Logout,
    "Logout", "Logout")
sessionMenu:Append(ID_InitPIN,
    "Init PIN", "Initialize PIN")
sessionMenu:Append(ID_SetPIN,
    "Set PIN", "Set PIN")
-- Data menu
-- Функции для работы с данными
local dataMenu = wx.wxMenu()
dataMenu:Append(ID_Data,
    "Data", "Data Processing")
dataMenu:Append(ID_Random,
    "Random", "Random Generator")
dataMenu:Append(ID_RandomSeed,
    "Random Seed", "Random Generator Seed")
dataMenu:Append(ID_RandEngine,
    "RAND Engine", "Load OpenSSL RAND Engine")
dataMenu:Append(ID_ASN1_Parse,
    "ASN.1 Parse",
    "Parse ASN.1 Structure to Tree Hierarchy")
-- PKCS#11 menu
-- Функции PKCS#11
local pkcs11Menu = wx.wxMenu()
pkcs11Menu:Append(ID_P11_Digest,
    "Digest",
    "Generate GOST R34.11-94, GOST R34.11-2012, SHA-1 or MD5
   Digest")
pkcs11Menu:Append(ID_P11_GenKeyPair,
    "Generate GOST Key Pair",
    "Generate GOST R34.10-2001 or GOST R34.10-2012 Key Pair")
pkcs11Menu:Append(ID_P11_CreatePubKey,
    "Create GOST Public Key",
    "Create GOST R34.10-2001 or GOST R34.10-2012 Public Key")
pkcs11Menu:Append(ID_P11_ImportCertificate,
    "Certificate Import",
```

```
    "Import X.509 Certificate Object")
pkcs11Menu:Append(ID_P11_ExportCertificate,
    "Certificate Export",
    "Export X.509 Certificate Object")
pkcs11Menu:Append(ID_P11_SignDigest,
    "GOST Sign Digest",
    "GOST R34.10-2001 or GOST R34.10-2012 Generate Digest
   Signature")
pkcs11Menu:Append(ID_P11_VerifyDigest,
    "GOST Verify Digest Signature",
    "GOST R34.10-2001 or GOST R34.10-2012 Verify Digest
   Signature")
pkcs11Menu:Append(ID_P11_SignData,
    "GOST Sign Data",
    "GOST R34.10-2001 or GOST R34.10-2012 Generate Data
   Signature")
pkcs11Menu:Append(ID_P11_VerifyData,
    "GOST Verify Data Signature",
    "GOST R34.10-2001 or GOST R34.10-2012 Verify Data Signature"
   )
pkcs11Menu:Append(ID_P11_GenKeyGost28147,
    "GOST 28147-89 Generate Key",
    "GOST 28147-89 Generate Key")
pkcs11Menu:Append(ID_P11_Encrypt,
    "GOST 28147-89 Encrypt",
    "GOST 28147-89 Encrypt Data")
pkcs11Menu:Append(ID_P11_Decrypt,
    "GOST 28147-89 Decrypt",
    "GOST 28147-89 Decrypt Data")
pkcs11Menu:Append(ID_P11_MacSign,
    "GOST 28147-89 MAC Sign",
    "GOST 28147-89 Generate MAC")
pkcs11Menu:Append(ID_P11_MacVerify,
    "GOST 28147-89 MAC Verify",
    "GOST 28147-89 Verify MAC")
pkcs11Menu:Append(ID_P11_CreateObject,
    "Create Object",
    "Create Object By Template")
pkcs11Menu:Append(ID_P11_CopyObject,
    "Copy Object",
    "Copy Object With New Template")
pkcs11Menu:Append(ID_P11_SetAttributes,
    "Set Attributes",
    "Set Object Attributes With Given Template")
```

```
pkcs11Menu:Append(ID_P11_RemoveAll,
    "Remove All Objects",
    "Remove all visible objects")
-- OpenSSL menu
-- Функции OpenSSL
local opensslMenu = wx.wxMenu()
opensslMenu:Append(ID_Digest,
    "Digest",
    "Generate GOST R34.11-94, GOST R34.11-2012, SHA-1 or MD5
   Digest")
opensslMenu:Append(ID_Encrypt,
    "GOST 28147-89 Encrypt",
    "GOST 28147-89 Encrypt Data")
opensslMenu:Append(ID_Decrypt,
    "GOST 28147-89 Decrypt",
    "GOST 28147-89 Decrypt Data")
opensslMenu:Append(ID_SignDigest,
    "GOST R34.10-2001 Sign Digest",
    "GOST R34.10-2001 Generate Digest Signature")
opensslMenu:Append(ID_VerifyDigest,
    "GOST R34.10-2001 Verify Digest Signature",
    "GOST R34.10-2001 Verify Digest Signature")
opensslMenu:Append(ID_SignDigest512,
    "GOST R34.10-2012-512 Sign Digest",
    "GOST R34.10-2012-512 Generate Digest Signature")
opensslMenu:Append(ID_VerifyDigest512,
    "GOST R34.10-2012-512 Verify Digest Signature",
    "GOST R34.10-2012-512 Verify Digest Signature")
opensslMenu:Append(ID_MacSign,
    "GOST 28147-89 MAC",
    "Generate GOST 28147-89 MAC")
opensslMenu:Append(ID_CSR,
    "GOST Certificate Signing Request",
    "Generate Certificate Signing Request")
opensslMenu:Append(ID_CSR_Sign,
    "CA Issue X509 Certificate",
    "CA Issue X509 Certificate For CSR")
opensslMenu:Append(ID_CertCheck,
    "Check X509 Certificate",
    "Check Validity And Purpose Of X509 Certificate")
opensslMenu:Append(ID_P12_Export,
    "Export To PKCS12",
    "Export Private Key And Certificate To PKCS12")
opensslMenu:Append(ID_P12_Import,
```

```
    "Import From PKCS12",
    "Import Private Key And Certificate From PKCS12")
opensslMenu:Append(ID_P7_Encrypt,
    "PKCS7 Encrypt",
    "Encrypt Data To PKCS7 Format")
opensslMenu:Append(ID_P7_Decrypt,
    "PKCS7 Decrypt",
    "Decrypt Data From PKCS7 Format")
opensslMenu:Append(ID_P7_Sign,
    "PKCS7 Sign",
    "Sign Data To PKCS7 Format")
opensslMenu:Append(ID_P7_Verify,
    "PKCS7 Verify",
    "Verify Signed Data From PKCS7 Format")
opensslMenu:Append(ID_HTTPS_Client,
    "HTTPS Client",
    "Simple HTTPS Client")
opensslMenu:Append(ID_Engine,
    "Load Engine",
    "Load Additional Engine And Make It Default")
-- Help menu
-- Информационные функции
local helpMenu = wx.wxMenu()
helpMenu:Append(ID_Manual,
    "Manual", "Show PDF Manual")
helpMenu:Append(wx.wxID_ABOUT,
    "About", "Show About Dialog")

-- Далее идут обработчики пунктов меню,
-- пример одного из которых приведен ниже.
-- ...

-- Обработчики событий назначены, завершаем построение главного
   меню
menuBar = wx.wxMenuBar()
menuBar:Append(fileMenu, "File")
menuBar:Append(slotsMenu, "Slot")
menuBar:Append(sessionMenu, "Session")
menuBar:Append(dataMenu, "Data")
menuBar:Append(pkcs11Menu, "PKCS#11")
menuBar:Append(opensslMenu, "OpenSSL")
menuBar:Append(helpMenu, "Help")

-- Назначаем главное меню главному диалогу
```

```
frame:SetMenuBar(menuBar)
```

11.2.1 Добавление пункта

Для добавления нового пункта нужно создать его уникальный идентификатор в скрипте wxid.lua, а затем оформить новый пункт, как это сделано для уже имеющихся пунктов.

```
opensslMenu:Append(ID_P7_Verify,
  "PKCS7 Verify",
  "Verify Signed Data From PKCS7 Format")
```

11.2.2 Обработчик пункта

Для того, чтобы пункт заработал, нужно определить для него функцию-обработчик, как это сделано для имеющихся пунктов. Ниже приведен пример обработчика для пункта меню OpenSSL/PKCS7 Verify.

```
function OnP7Verify(evt)
-- Загрузка подписанных данных
    local ld = LuaData.new('Load PKCS#7 Signed Data', nil)
    local p7 = openssl.pkcs7_read(ld.data)
-- Загрузка доверенного сертификата УЦ
    local ld = LuaData.new('Load Trusted CA Certificate', nil)
    local recip_sk_x509 = openssl.sk_x509_read_data(ld.data)
    wx.wxBeginBusyCursor()
    local res = openssl.pkcs7_verify(p7, 0, nil, recip_sk_x509)
    wx.wxEndBusyCursor()
    if res == true then
        wx.wxLogMessage("PKCS#7 verification OK")
    else
        wx.wxLogError("PKCS#7 verification failed")
    end
end
frame:Connect(ID_P7_Verify, wx.wxEVT_COMMAND_MENU_SELECTED,
    OnP7Verify)
```

11.3 Функциональные модули

Для создания функциональных объектно-ориентированных классов в утилите организовано единообразное оформление Lua-модулей. Например, диалоговая обработка значения атрибута объекта PKCS#11 производится в модуле Attribute.lua.

Все модули в утилите начинаются более или менее единообразно. В начале модуля Attribute.lua содержатся строки, определяющие конструирование объекта данного класса Attribute. Затем идут прикладные функции, вызываемые из конструктора.

Листинг 11.1: Attribute.lua

```
-- Copyright (C) ООО "ЛИССИ-Софт", 2012-2014
local base = _G
local Factory = require("Factory")
local wx = require("wx")
local funcs = require("luacryptoki")
local string = string
local table = table
local LuaData = require("LuaData")

module("Attribute")

meta = {__index = _M}

-- Прокси-конструктор
function new(attr)
    return Factory.create(_M, attr)
end

-- Диалоговый конструктор
function construct(self, attr)
    self.type = attr.type
    self.pValue = attr.pValue
    self:CreateDialog()
    self:ShowDialog()
    self:DestroyDialog()
end

function CreateDialog(self)
    local title = "Attribute"
    dlg = wx.wxDialog(wx.NULL, wx.wxID_ANY, title,
        wx.wxDefaultPosition, wx.wxDefaultSize,
        wx.wxSTAY_ON_TOP)
  local topSizer = wx.wxBoxSizer(wx.wxVERTICAL)
  -- Назначаем сайзер диалогу
  dlg:SetSizer(topSizer)
  -- Внутренний сайзер с рамкой и заголовком
  local sizer = wx.wxStaticBoxSizer(wx.wxVERTICAL, dlg, title)
    -- Выбор атрибута
    combo = wx.wxComboBox(dlg, wx.wxID_ANY, "Attribute Type",
```

```
        wx.wxDefaultPosition, wx.wxDefaultSize,
        base.AttrNames)
    combo:SetValue(base.Attr[self.type].name)
  sizer:Add(combo, 0, wx.wxALL + wx.wxALIGN_CENTER, 10)
    -- Окно данных
  text_ctrl = wx.wxTextCtrl(dlg, wx.wxID_ANY, "",
    wx.wxDefaultPosition, wx.wxSize(360, 200),
        wx.wxTE_MULTILINE + wx.wxHSCROLL)
  sizer:Add(text_ctrl, 0, wx.wxALL + wx.wxALIGN_CENTER, 10)
    -- Горизонтальный сайзер кнопок
  local horSizer = wx.wxBoxSizer(wx.wxHORIZONTAL)
    -- Кнопка "Data"
  local button_data = wx.wxButton(dlg, base.ID_Data, "Data")
    -- Пока что кнопка "Data" работает только для строк байтов.
    -- Для других типов атрибутов требуются соответствующие
    -- доработки.
    if base.type(self.pValue) == "string" then
        dlg:Connect(base.ID_Data,
 wx.wxEVT_COMMAND_BUTTON_CLICKED,
            function (evt) self:OnData(evt) end)
    else
        button_data:Enable(false)
    end
  horSizer:Add(button_data, 0, wx.wxALL + wx.wxALIGN_CENTER, 10)

    -- Кнопка "OK"
  local button_ok = wx.wxButton(dlg, wx.wxID_OK, "OK")
  button_ok:SetDefault()
  horSizer:Add(button_ok, 0, wx.wxALL + wx.wxALIGN_CENTER, 10)
    -- Кнопка "Cancel"
  local button_cancel =
        wx.wxButton(dlg, wx.wxID_CANCEL, "Cancel")
  horSizer:Add(button_cancel, 0, wx.wxALL + wx.wxALIGN_CENTER,
   10)

    sizer:Add(horSizer, 0, wx.wxALL + wx.wxALIGN_CENTER, 10)

  topSizer:Add(sizer, 0, wx.wxALL + wx.wxALIGN_CENTER, 10)

  -- Минимализация размера диалога
  topSizer:Fit(dlg)
    local attr = {type = self.type, pValue = self.pValue}
    local text = base.value2str(attr, true)
    text_ctrl:SetValue(text)
```

```
end

function OnData(self, evt)
    local ld = LuaData.new("Attribute Data", self.pValue)
    local attr = {type = self.type, pValue = ld.data}
    local text = base.value2str(attr, true)
    text_ctrl:SetValue(text)
    ld:destroy()
end

function ShowDialog(self)
    if dlg:ShowModal() == wx.wxID_OK then
        local attr_type = base.FindAttrByName(combo:GetValue())
        local text = text_ctrl:GetValue()
        self.type = attr_type
        self.pValue = base.str2value(self.type, text)
        -- Исходный атрибут здесь не изменяется!
        -- Корректность значений здесь пока не проверяется,
        -- а надо бы!
        -- Для проверки корректности следует использовать
        -- паттерны регулярных выражений.
    end
end

function DestroyDialog(self)
    dlg:Destroy()
end
```

12 Ссылки

1. Официальный сайт ООО "ЛИССИ-Софт". – http://soft.lissi.ru/.

2. Технический комитет по стандартизации "Криптографическая защита информации" ТК 26. – 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 34.11-94. Информационная технология. Криптографическая защита информации. Функция хеширования. – http://protect.gost.ru/document.aspx?control=7&id=134550.

7. ГОСТ Р 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хеширования. – Москва, Стандартинформ, 2012.

8. 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. – CRYPTO-PRO, January 2006. – http://www.ietf.org/rfc/rfc4357.txt.

9. RFC 4490. S. Leontiev, Ed., G. Chudov, Ed. 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). – CRYPTO-PRO, May 2006. – http://tools.ietf.org/html/rfc4490.

10. Методические рекомендации по заданию узлов замены блока подстановки алгоритма шифрования ГОСТ 28147-89 (готовится к публикации). – Москва, ТК 26, 2013.

11. Методические рекомендации по заданию параметров эллиптических кривых в соответствии с ГОСТ Р 34.10-2012 (готовится к публикации). – Москва, ТК 26, 2013.

12. Методические рекомендации по криптографическим алгоритмам, сопутствующим применению стандартов ГОСТ Р 34.10-2012 и ГОСТ Р 34.11-2012 (готовится к публикации). – Москва, ТК 26, 2013.

13. Парольная защита с использованием алгоритмов ГОСТ. Дополнения к PKCS#5 (готовится к публикации). – Москва, ТК 26, 2013.

14. Расширение PKCS#11 для использования российских криптографических алгоритмов. – Технический комитет по стандартизации (ТК 26) "Криптографическая защита информации". – Москва, ТК 26, 2008.

15. 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.

16. Официальный сайт проекта OpenSSL. – http://www.openssl.org/.

17. PKCS#11 v2.30: Cryptographic Token Interface Standard. – RSA Laboratories, 2009. – http://www.rsa.com/rsalabs/node.asp?id=2133.

18. wxLua - wxWidgets bindings for Lua. – http://wxlua.sourceforge.net.

19. wxWidgets - Cross-Platform GUI Library. – http://www.wxwidgets.org.

20. The Programming Language Lua. – http://www.lua.org.

21. Zhaozg lua-openssl. – https://github.com/zhaozg/lua-openssl.

22. Network support for the Lua language. – http://w3.impa.br/~diego/software/luasocket/.