«Крипто-Веб»
версия 1.0.0
Руководство программиста

ООО «ЛИССИ-Софт»

13 сентября 2013 г.

Оглавление

1 Общие сведения

«Крипто-Веб»представляет собой платформонезависимый плагин, функционирующий на различных операционных системах (MS Windows, Linux, OS X), и использующийся в различных браузерах – Internet Explorer, Mozilla Firefox, Safari, Opera, Google Chrome, Seamonkey. Будучи внедренным на страницу, плагин предоставляет web-скрипту доступ к следующим высокоуровневым криптографическим функциям:

- Формирование и проверка электронной подписи по стандарту PKCS#7;
- Шифрование данных на открытом ключе по стандарту PKCS#7;
- Извлечение исходных данных из зашифрованных контейнеров PKCS#7 и контейнеров с присоединенной подписью;
- Установка защищенного соединения с сервером по протоколу TLS;
- Формирование запроса на сертификат в формате PKCS#10;
- Просмотр сертификатов;
- Импорт/экспорт сертификатов в формате PKCS#12;
- Расшифровка почтовых сообщений SMIME.

Назначением данного продукта является обеспечение поддержки российских криптографических алгоритмов как в новых, так и уже существующих решениях, построенных на базе Web-технологий. Предполагается, что плагин найдет применение в системах документооборота, системах дистанционного банковского обслуживания, на порталах госуслуг и т.п.

Для своей работы плагин LCSignPlugin может использовать одну из высокоуровневых библиотек:

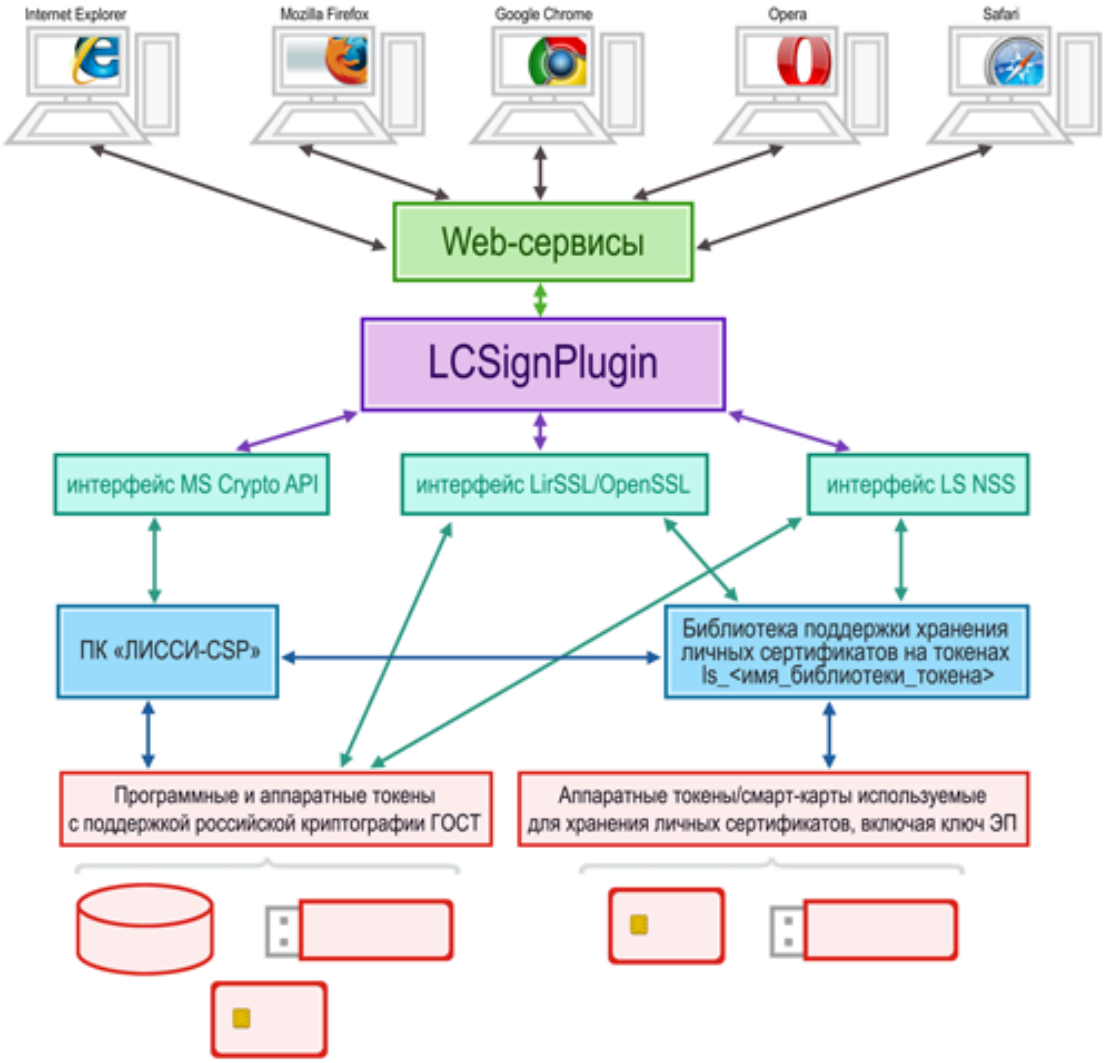

Рис. 1.1

- NSSCryptoWrapper;

  Библиотека NSSCryptoWrapper написана с использованием LS NSS (Network Security Services с поддержкой российской криптографии) и подразумевает использование в качестве СКЗИ программых [1] и аппаратных токенов/смарт-карт, поддерживающих российскую криптографию (ruToken [2], eToken [3], MSKey-K [4] и другие) и имеющих интерфейс PKCS#11.

- CSPCryptoWrapper;

  Библиотека CSPCryptoWrapper использует CryptoAPI MS Windows и используется только на платформе MS Windows. В качестве базового СКЗИ она предполагает использование MS CSP с поддержкой российской криптографии, на-

пример «ЛИССИ-CSP» [5].

- LirCryptoWrapper;

  Библиотека LirCryptoWrapper написана на базе кроссплатформенной программной библиотеки защиты информации (ПБЗИ «СКЗИ ЛИРССЛ») [6].

- SSLCryptoWrapper.

  Библиотека SSLCryptoWrapper написана на базе OpenSSL 1.0 с подключаемым engine lccore [7].

Все библиотеки имеют один и тот же пользовательский интерфейс.

Примеры использования плагина можно найти в соответствующем разделе сайта ООО «ЛИССИ-Софт»[8] – исходный код каждой страницы отформатирован, чтобы служить не только средством тестирования плагина, но и примером исходного кода, раскрывающего некоторый аспект использования продукта.

2 Основные понятия и принципы работы плагина

Ниже приводятся основные понятия, использующиеся при описании работы плагина:

- **Локальная база данных плагина** – набор файлов, размещающийся в домашнем каталоге пользователя, содержащий сведения о подключенных модулях внешних СКЗИ, а также могущий служить хранилищем для корневых сертификатов. Если при запуске плагина локальная база данных отсутствует, она создается автоматически.

- **Токен** – динамически подключаемый модуль, имеющий библиотеку интерфейса стандарта PKCS#11, предоставляющий одновременно как механизмы выполнения криптографических преобразований (т.е. функционирующий как внешний СКЗИ), так и механизмы хранения сертификатов и ключевого материала. Перед использованием токен должен пройти процедуру регистрации в локальной базе данных плагина, после чего становится доступным использование его механизмов и объектов, хранящихся на нем. Если для получения доступа к закрытым объектам токена требуется ввод PIN-кода, он будет автоматически запрошен у пользователя. Помимо реальных токенов в плагине всегда имеется два виртуальных: `all` и `internal`. Токен `all` соответствует совокупности всех подключенных к плагину модулей. Например, если необходимо вывести список всех сертификатов во всех хранилищах, доступных плагину, следует в качестве имени токена указать `all`. Токен `internal` представляет собой внутреннее хранилище сертификатов плагина. Этот модуль имеет следующие особенности: во-первых, он не способен хранить закрытые ключи, и таким образом импорт личных сертификатов пользователя в него невозможен; во-вторых, при импорте в плагин корневого сертификата Удостоверяющего Центра его копия всегда сохраняется на токене `internal`.

При регистрации в базе данных плагина подключенным токенам и импортируемым сертификатам присваивается внутреннее имя – т.н. `nickname`. В дальнейшем `nickname` используется для идентификации токена или сертификата при вызове функций, предоставляемых плагином.

Все функции следуют следующим соглашениями, если специально не оговорено иное:

- Функции типа `int` в случае успешного выполнения возвращают значение `0` и `-1` в случае сбоя.

- Функции типа `Array` в случае сбоя вместо предписанного значения возвращают массив нулевой длины.
- Функции типа `String` в случае сбоя вместо предписанного значения возвращают пустую строку.

Плагин ведет журнал сообщений об ошибках, полезный при отладке.

В Windows отладочные сообщения сохраняются в каталоге `C:\Temp\nss_infolog.txt`, если каталог `Temp` отсутствует, он не создается автоматически.

В Unix журнал сохраняется в `/var/tmp/nss_infolog.txt`.

3 Описание предоставляемых функций

3.1 Вспомогательные функции

`String SelectDir()`
Отображает стандартный системный диалог выбора директории.
Возвращает полный путь до выбранного пользователем каталога.

`String SelectFile()`
Отображает стандартный системный диалог выбора файла.
Возвращает полное имя выбранного пользователем файла.

`objects About()`
Возвращает информацию о плагине в виде объекта со свойствами:

- `string need_token` – равен «false», если отсутствует библиотека NSSCryptoWrapper;
- `string wrapper_type` – тип интерфейса PKCS#11;
- `string about` – информация о разработчике;
- `string copyright` – информация о правообладателе.

`string ReadFile (String Filename)`
Метод позволяет получить содержимое файла **Filename** в виде строки.

3.2 Работа с внешними СКЗИ (токенами)

`int AddModulePKCS11(String Module, String Library)`
Регистрирует во внутренней базе данных новый модуль под именем `Module`, работающий через подключаемую библиотеку, находящуюся в файле `Library`.

`Array ListToken()`

Функция возвращает массив имен всех подключенных к плагину токенов. Помимо модулей, подключенных через функцию `AddModulePKCS11`, функция всегда возвращает виртуальные токены `all` и `internal`.

`int P12ExportFile(String Module, String Nickname, String OutFile)`
Метод позволяет экспортировать сертификат с внутренним именем **Nickname** из хранилища сертификатов **Module** в файл формата PKCS#12. Параметр **OutFile** указывает на каталог, в который будет сохранен файл сертификата.

`string GetCertContent (String Nickname)`
Возвращает сертификат с внутренним именем **Nickname** в формате PEM.

`string CertOrReqView(String NicknameOrFile, bool isCert)`
Метод возвращает содержимое сертификата с внутренним именем **NicknameOrFile** в случае, если параметр **isCert** равен **true**, или содержимое запроса PKCS#10, находящегося в файле **NicknameOrFile**, если параметр **isCert** равен **false**.

3.3 Работа с сертификатами

`Array PrivCert(String Module)`
Возвращает список, по одному имени на строку, личных сертификатов (таких, для которых имеется закрытый ключ), хранящихся на подключенном токене `Module`. Если указан виртуальный токен `all`, то плагин последовательно переберет все подключенные к нему модули. необходимости у пользователя ввод PIN-кода для каждого токена. Вызов данной функции с указанием виртуального токена `internal` хотя и возможен, но результатом всегда будет пустой массив.

`Array AllCert(String Module)`
Возвращает список, по одному имени на строку, всех сертификатов, хранящихся на подключенном токене `Module`. Следует отметить, что помимо сертификатов, имеющих только открытый ключ (сторонних), функция также возвращает и список сертификатов, для которых имеется закрытый ключ (личных). Таким образом, множество значений функции `AllCert` включает в себя и множество значений функции `PrivCert`. Указание токена `all` приведет к перебору сертификатов на всех подключенных к плагину модулей, указание `internal` - к перебору сертификатов, хранящихся во внутренней базе данных плагина.

`Array GetCertInfo(String Nickname)`
Возвращает значение полей сертификата (по одному на строку), имеющего внутреннее имя `Nickname`.

`String AddCert(String Module, String Trusts, String Content, bool IsFile)`
Функция предназначена для импорта сертификата на один из токенов (обозначенный параметром `Module`), подключенных к плагину.

Если параметр `IsFile` имеет значение `true`, то параметр `Content` должен содержать имя файла, в котором хранится сертификат в формате DER либо PEM. Если же `IsFile` равен `false`, то параметр Content воспринимается как строка, содержащая сертификат, закодированный в формате Base64 либо PEM.

Параметр `Trusts` - строка, задающая параметры использования для импортируемого сертификата. Она состоит из трех полей, разделенных запятыми: «`x,y,z`». Поле `x` отвечает за использование сертификата при установлении защищенного соединения по протоколу SSL/TLS, поле `y` – за применение сертификата при отправке защищенных почтовых сообщений (S/MIME), параметр `z` определяет применение сертификата при подписании произвольных данных (документов).

Каждое из трех полей может содержать следующие флаги:

- `p` – использование сертификата запрещено;
- `P` – доверенная удаленная сторона. Данный сертификат всегда считается действительным, даже при отсутствии корневого сертификата УЦ;
- `c` – корневой сертификат Удостоверяющего Центра. Сертификат будет использован для проверки некорневых сертификатов;
- `T` – доверенный Удостоверяющий Центр, имеющий право выпуска только клиентских сертификатов. Сертификат сервера SSL, выпущенный на данном УЦ, будет считаться недействительным;
- `C` – доверенный Удостоверяющий Центр, имеющий право выпуска только серверных сертификатов. Действие флага аналогично действию флага `T`, но в отношении клиентских сертификатов;
- `u` – сертификат пользователя. Сертификат подвергается стандартному набору проверок, прежде чем считаться действительным.

Как правило, используется два типовых набора флагов: «`u,u,u`» для сертификата стороннего пользователя и «`CT,CT,CT`» для корневого сертификата УЦ.

Следует иметь в виду, что при импорте корневого сертификата его копия всегда сохраняется на виртуальном токене `internal`. Таким образом, иимпорт корневого сертификата на токен, отличный от `internal`, повлечет за собой появление двух одинаковых зарегистрированных копий сертификата.

Если импорт сертификата прошел успешно, функция возвращает внутреннее имя, присвоенное сертификату.

```
int P12Import(String Module, String File)
```

Функция выполняет импорт личного сертификата на токен `Module` с восстановлением закрытого ключа из контейнера PKCS#12, хранящегося в файле `File`. Если указан токен `all`, то будет выбран первый подходящий токен. Импорт на токен `internal` невозможен. При импорте у пользователя будет запрошен пароль от контейнера PKCS#12 и, при необходимости, PIN-код токена, на который производится импорт.

`int DeleteCert(String Nickname, bool IsPrivate)`

Удаляет сертификат с внутренним именем `Nickname`. При удалении личного сертификата, если параметр `IsPrivate` имеет значение `true`, удаляется также и соответствующий закрытый ключ. Если `IsPrivate` имеет значение `false`, закрытый ключ останется на токене. В таком случае, при повторном импорте сертификата функцией `AddCert` на тот же токен, закрытый ключ вновь связывается с сертификатом и таким образом последний приобретает статус личного.

3.4 Работа с электронной подписью

`int P7SignFile(String Nickname, String InFile, String OutFile, bool IsAttached)`

Выполняет подписание файла `InFile` с помощью **личного** сертификата `NickName`. Результат – подпись в формате PKCS#7 – сохраняется в файле `OutFile`. Если значение параметра `IsAttached` равно `true`, то создается присоединенная подпись – контейнер PKCS#7 содержит не только электронную цифровую подпись, но и исходные данные целиком. Если же `IsAttached` равен `false`, то выходной файл будет содержать только подпись, без исходных данных.

`int P7SignText(String Nickname, String InText, String OutText, bool IsAttached)`

Функция аналогична `int P7SignFile`, однако работает не с файлами, а с текстовыми строками. Подписанию подвергается текст, содержащийся в строке `InText`, результат в виде контейнера PKCS#7, закодированного в Base64, сохраняется в `OutText`.

`Array P7VerifyFile(String Content, String Signature)`

Функция выполняет проверку электронной цифровой подписи. Исходные данные берутся из файла `Content`, подпись – из файла `Signature`. Если проверка подписи прошла успешно, первый элемент массива содержит дату подписания данных в текстовом формате, а второй – сертификат подписанта, закодированный в Base64. Импорт сертификата подписанта в локальную базу данных можно выполнить с помощью функции `AddCert`.

`Array P7VerifyText(String Content, String Signature)`

Функция аналогична `P7VerifyFile`. Параметр `Content` должен содержать исходный текст, параметр `Signature` – закодированный в Base64 контейнер PKCS#7, содержащий подпись.

3.5 Шифрование данных

`int P7EnvFile(String Nickname, String InFile, String OutFile)`

Выполняет шифрование данных. Параметр `Nickname` задает сертификат получателя, параметры `InFile` и `OutFile` - имена входного и выходного файлов соответственно.

`int P7EnvText(String Nickname, String InText, String OutText)`

Аналог `P7EnvFile`, работающий со строками. В случае успешного выполнения `OutText` будет содержать зашифрованный контейнер PKCS#7, закодированный в Base64.

3.6 Извлечение данных из контейнера PKCS#7

`int P7ContentFile(String Container, String Content)`

Извлекает исходные данные из зашифрованного контейнера или контейнера с присоединенной подписью PKCS#7. Параметр `Container` задает имя файла, содержащего контейнер, `Content` – имя файла, в который будет записано извлеченное содержимое.

`String P7ContentText(String Container)`

Аналог `P7ContentFile`, работающий со строками. Параметр `Container` должен содержать контейнер PKCS#7, закодированный в Base64. При успешном выполнении функция возвращает извлеченный из контейнера текст.

`string P7EnvTextSender(String SenderNickname, String GetterNickname, String Content)`

Результатом выполнения данного метода является строка **Content**, являющаяся результатом асимметричного шифрования с помощью открытого ключа сертификата **SenderNickname** и закрытого ключа сертификата **GetterNickname** в формате BASE64.

3.7 Установление защищенного соединения по TLS

`String TLSClient(String Nickname, String Host, int Port, String Data)`

Данная функция позволяет отправлять и принимать данные по защищенному протоколу TLS с поддержкой российской криптографии. Открывается защищенное соединение с сервером `Host` на порте `Port`. Затем серверу предоставляется сертификат указанный в параметре `Nickname`, после чего устанавливается авторизованное соединение. Параметр `Nickname` может являться пустой строкой («») – в таком случае устанавливается анонимное соединение. Затем функция отправляет на сервер данные, содержащиеся в параметре `Data` и ждет ответа от сервера. После получения ответа соединение разрывается. В случае успешного выполнения возвращается строка, содержащая данные, полученные от сервера.

3.8 Формирование запроса на сертификат

`int CreateCertReq(String Token, String Subject, String KeyUsage, String OutFile, int KeyParams, bool IsPem)`

Функция позволяет сгенерировать ключевую пару и создать запрос на сертификат в формате PKCS#10 с последующим сохранением результата в файл. Параметр `Token` задает, на каком из подключенных токенов будет сгенерирована ключевая пара, а `KeyParams` – какой набор криптопараметров будет использован. Ниже приведен список соответствия значения параметра `KeyParams` наборам криптопараметров:

- `0` – GostR3410-2001-CryptoPro-A-ParamSet;
- `1` – GostR3410-2001-CryptoPro-B-ParamSet;
- `2` – GostR3410-2001-CryptoPro-C-ParamSet;
- `3` – GostR3410-2001-CryptoPro-XchA-ParamSet;
- `4` – GostR3410-2001-CryptoPro-XchB-ParamSet.

Параметр `KeyUsage` задает в виде текстовой строки область использования ключа:

- `digitalSignature` – цифровая подпись;
- `nonRepudiation` – неотрекаемость;
- `keyEncipherment` – шифрование ключей (используется при взаимодействии по протоколу SSL/TLS);
- `dataEncipherment` – шифрование данных;
- `certSigning` – подписание издаваемых сертификатов;
- `crlSigning` – подписание издаваемых списков отзыва сертификатов (CRL);
- `critical` – сигнализирует о том, что область применения ключа критична.

Эти значения могут быть перечислены в строке через запятую, например:

```
"digitalSignature, nonRepudiation, critical"
```

`Subject` задает значения поля для владельца запроса, например:

```
CN=Иванов Иван Иванович, C=RU, O=Организация, OU=Бухгалтерия
```

`OutFile` – имя выходного файла, в который будет сохранен запрос на сертификат.
`IsPem` – если значение параметра `true`, то запрос будет сохранен в формате PEM, если `false` – в формате DER.

3.9 Расшифровка и извлечение содержимого SMIME файла

`int SMIMEContentFile(String SMIMEFile, String OutFile)`

Данный метод позволяет расшифровать и сохранить содержимое SMIME-сообщения, которое находится в файле **SMIMEFile**, в файл **OutFile**.

4 Ссылки

1. http://soft.lissi.ru/products/skzi/PKCS11/
2. http://www.rutoken.ru
3. http://www.aladdin-rd.ru/catalog/etoken/
4. http://multisoft.ru/katalog/zawita_informacii/skzi_ms_key_k/usb-_klyuch_ms_key_k/
5. http://soft.lissi.ru/products/skzi/lissi-csp/
6. http://soft.lissi.ru/products/skzi/lirssl/
7. http://soft.lissi.ru/products/skzi/OpenSSL/
8. http://soft.lissi.ru/products/crypto-web/nss_plugin_examples/