ViPNet PKI Client

Руководство разработчика

infotecs

ФРКЕ.00175-01 33 01

Версия продукта 1.5.1

ОАО «ИнфоТеКС»

127287, г. Москва, Старый Петровско-Разумовский проезд, д. 1/23, стр. 1, 2 этаж

Телефон: +7 (495) 737-6192, 8-800-250-0260 — бесплатный звонок из России (кроме Москвы)

Веб-сайт: infotecs.ru

Служба технической поддержки: hotline@infotecs.ru

Содержание

Введение

О документе

В документе содержится информация о компоненте SDK, который необходим для создания веб-приложений, обращающихся для выполнения криптографических операций к программе Web Unit. Компонент SDK представляет собой комплект средств разработки на языке JavaScript.

Для кого предназначен документ

Руководство предназначено для JavaScript-разработчиков, которые используют функции комплекта средств разработки SDK, описанные в данном документе.

Предполагается, что читатель этого руководства знаком с языком JavaScript и имеет представление о базовых понятиях в области криптографии, а также об инфраструктуре открытых ключей PKI (см. глоссарий, стр. 80).

Соглашения документа

Ниже перечислены соглашения, принятые в этом документе для выделения информации.

Таблица 1. Обозначения, используемые в примечаниях

Обозначение	Описание
	**Внимание!** Указывает на обязательное для исполнения или следования действие или информацию.
	**Примечание.** Указывает на необязательное, но желательное для исполнения или следования действие или информацию.
	**Совет.** Содержит дополнительную информацию общего характера.

Таблица 2. Обозначения, используемые для выделения информации в тексте

Обозначение	Описание
Название	Название элемента интерфейса. Например, заголовок окна, название поля, кнопки или клавиши.
Клавиша+Клавиша	Сочетание клавиш. Чтобы использовать сочетание клавиш, следует нажать первую клавишу и, не отпуская ее, нажать вторую клавишу.
Меню > Подменю > Команда	Иерархическая последовательность элементов. Например, пункты меню или разделы на панели навигации.
`Код`	Имя файла, путь, фрагмент текстового файла (кода) или команда, выполняемая из командной строки.

О программе

В состав ПК ViPNet PKI Client входит комплект средств разработки SDK. С помощью этого комплекта разработчики могут встраивать в веб-приложения вызов таких криптографических операций, как заверение данных электронной подписью, проверка электронной подписи, шифрование и расшифрование данных, а также создание запроса на сертификат и установка полученного сертификата в системное хранилище. Комплект средств разработки SDK при работе взаимодействует с криптопровайдером ViPNet CSP и программой Web Unit.

Обратная связь

Дополнительная информация

Сведения о продуктах и решениях ViPNet, распространенные вопросы и другая полезная информация собраны на сайте ОАО «ИнфоТеКС»:

- Информация о продуктах ViPNet.
- Информация о решениях ViPNet.
- Часто задаваемые вопросы.
- Форум пользователей продуктов ViPNet.

Контактная информация

Если у вас есть вопросы, свяжитесь со специалистами ОАО «ИнфоТеКС»:

- Единый многоканальный телефон:

  +7 (495) 737-6192,

  8-800-250-0-260 — бесплатный звонок из России (кроме Москвы).

- Служба технической поддержки: hotline@infotecs.ru.

  Форма для обращения в службу технической поддержки через сайт.

  Консультации по телефону для клиентов с расширенной схемой технической поддержки: +7 (495) 737-6196.

- Отдел продаж: soft@infotecs.ru.

Если вы обнаружили уязвимости в продуктах компании, сообщите о них по адресу security-notifications@infotecs.ru. Распространение информации об уязвимостях продуктов ОАО «ИнфоТеКС» регулируется политикой ответственного разглашения.

1

Общие сведения

Назначение компонентов SDK и Web Unit

Стандартные веб-браузеры не поддерживают выполнение криптографических операций в веб-приложениях с использованием российских алгоритмов ГОСТ. Компонент SDK позволяет встраивать криптографические функции в веб-приложения, после чего пользователи с помощью программы Web Unit, работающей на их компьютерах в фоновом режиме, смогут выполнять следующие криптографические операции на веб-страницах:

- Создание запроса на сертификат.
- Установка сертификата в системное хранилище.
- Заверение файлов электронной подписью.
- Проверка электронной подписи.
- Шифрование файлов.
- Расшифрование файлов.
- Вычисление хэш-суммы.

Для шифрования и расшифрования файлов применяется асимметричный алгоритм: используется пара ключей, состоящая из закрытого и открытого ключей. Чтобы подтвердить принадлежность открытого ключа его владельцу, используется сертификат открытого ключа, выданный удостоверяющим центром (см. глоссарий, стр. 81).

Для формирования и проверки электронной подписи используется пара ключей, состоящая из закрытого и открытого ключей. Далее в документе эти ключи называются, соответственно, ключ электронной подписи (см. глоссарий, стр. 80) и ключ проверки электронной подписи (см. глоссарий, стр. 80). Чтобы подтвердить принадлежность ключа проверки электронной подписи его владельцу, используется сертификат ключа проверки электронной подписи (см. глоссарий, стр. 81), выданный удостоверяющим центром.

Примечание. Несмотря на то, что данные заверяются с помощью ключа электронной подписи, а шифрование выполняется с помощью открытого ключа получателя, в профессиональной речи используются выражения «подписать с помощью сертификата», «зашифровать с помощью сертификата».

Использование криптопровайдера в системах защиты данных

Для выполнения криптографических операций программа Web Unit обращается к криптопровайдеру ViPNet CSP.

Криптопровайдер — это специализированный программный модуль, позволяющий защищать данные средствами криптографии. При обращении со стороны Web Unit криптопровайдер ViPNet CSP выполняет следующие операции:

- Формирование и проверка электронной подписи в соответствии со стандартами ГОСТ Р 34.11–94, ГОСТ Р 34.11-2012, ГОСТ Р 34.10–2001 и ГОСТ Р 34.10–2012.
- Шифрование информации в соответствии с ГОСТ 28147–89.

Порядок выполнения криптографических операций с использованием программы Web Unit

Для вызова криптографических операций необходимо создать веб-приложение, обращающееся к программе Web Unit с помощью JavaScript-функций из комплекта средств разработки SDK, разместить его на веб-портале и обеспечить доступ пользователей веб-приложения к этому веб-порталу.

Рисунок 1. Взаимодействие программы Web Unit с веб-браузером

Криптографические операции выполняются в следующем порядке:

1. Пользователь в своем веб-браузере переходит на страницу стороннего веб приложения, находящуюся на удаленном веб-портале.
2. Пользователь задает на странице стороннего веб-приложения необходимые параметры и вызывает криптографическую операцию. При этом веб-браузер обращается к программе Web Unit, установленной на компьютере пользователя и работающей в фоновом режиме.
3. В зависимости от вызываемой криптографической операции появляются различные окна программы Web Unit.

   После указания всех запрашиваемых данных в окнах программа Web Unit обращается к криптопровайдеру ViPNet CSP для выполнения криптографической операции и получает результат.

Примечание. Вы можете также реализовать в своем веб-приложении вызов криптографических операций в режиме «Без подтверждения» (см. Добавление вызова криптографических функций в режиме «Без подтверждения» на стр. 15), при этом операция будет выполняться по запросу автоматически без подтверждения.

4 Программа Web Unit передает результат криптографической операции в веб-браузер.

5 Веб-браузер передает результат криптографической операции на веб-портал.

Добавление вызова криптографических функций в режиме «Без подтверждения»

В своих веб-приложениях вы можете использовать криптографические функции SDK не только для добавления на веб-сайт таких операций, как шифрование или формирование электронной подписи, но и для выполнения других действий, например организации системы аутентификации пользователей на основе их сертификатов. В таком случае вы можете использовать режим «Без подтверждения» (Bypass).

В режиме «Без подтверждения» не появляются окна программы ViPNet PKI Client Web Unit и операции выполняются автоматически.

Примечание. При выполнении криптографической операции может появиться окно криптопровайдера, в котором запрашивается пароль к контейнеру ключей, соответствующему выбранному сертификату.

При первом вызове операции в режиме «Без подтверждения» появится окно с запросом на выполнение операции. Если установить флажок **Больше не спрашивать для этого сайта** (в ОС Windows) или **Запомнить выбор для данного ресурса** (в ОС Linux), данное окно для этого сайта более не появится.

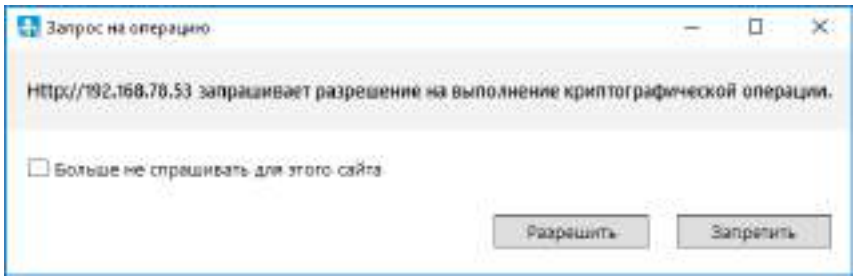

Рисунок 2. Запрос на выполнение криптографической операции (ОС Windows)

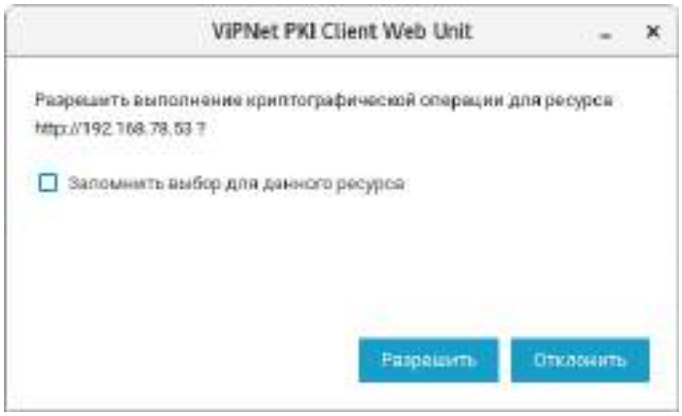

Рисунок 3. Запрос на выполнение криптографической операции (ОС Linux)

Если веб-сайт при обращении к программе ViPNet PKI Client Web Unit не указывает свой IP-адрес, то операция в режиме «Без подтверждения» будет автоматически отклонена.

Для вызова криптографических функций в режиме «Без подтверждения» необходимо инициализировать объект `LssClient` специальным образом (см. Добавление вызова криптографических функций в веб-приложения на стр. 21), а для выполнения функций `sign` и `signAndEncrypt` предварительно задать сертификат для электронной подписи с помощью функции `selectCertificate` (см. Функция selectCertificate на стр. 54).

Внимание! В режиме «Без подтверждения» нельзя вызывать функции generateCertificateRequest и installCertificate.

2

Работа с комплектом средств разработки ViPNet PKI Client SDK

Использование примеров кода вызова криптографических операций при написании веб-страниц

Для работы с комплектом SDK необходимо выполнить полную установку ViPNet PKI Client. Подробнее см. документ «ViPNet PKI Client. Общие сведения».

При полной установке ViPNet PKI Client в папку с программой копируются следующие примеры веб-приложений:

- Пример вызова криптографических операций при передаче в качестве входных параметров бинарных данных (на стр. 18).
- Пример вызова криптографических операций при передаче в качестве входных параметров файлов (на стр. 19).

Вы можете использовать эти примеры для создания собственных веб-приложений, вызывающих криптографические операции с помощью программы Web Unit.

Функции, приведенные в примерах, подробно описаны в разделе Добавление вызова криптографических функций в веб-приложения (на стр. 21).

Пример вызова криптографических операций при передаче в качестве входных параметров бинарных данных

Пример представляет собой веб-приложение, на страницах которого пользователь может вызывать криптографические функции, указывая в качестве входных параметров бинарные данные. Для ОС Windows пример находится в следующей папке:

- `C:\Program Files\InfoTeCS\ViPNet PKI Client\SDK\Sites\Base64DataOnly` — для 32-разрядных ОС.
- `C:\Program Files (x86)\InfoTeCS\ViPNet PKI Client\SDK\Sites\Base64DataOnly` — для 64-разрядных ОС.

Для ОС Linux пример находится в каталоге `/opt/itcs/share/pki-client/Sites/Base64`.

В примере в качестве входных параметров для выполнения криптографических операций используются бинарные данные в формате Base64. Для перевода данных в этот формат вы можете использовать онлайн-конвертеры).

При необходимости представления сертификата в формате Base64 предварительно экспортируйте его в этот формат.

Пример вызова криптографических операций при передаче в качестве входных параметров файлов

Пример представляет собой веб-приложение, на страницах которого пользователь может вызывать криптографические функции, передавая в качестве входных параметров конкретные файлы.

В ОС Windows работа с файлами в примере реализована с помощью тестового файл-сервера, развернутого на компьютере разработчика. Программу тестового файл-сервера необходимо запустить после установки ПО. Она работает в фоновом режиме аналогично программе Web Unit.

Примечание. На практике при реализации работы с файлами файл-сервер необходимо расположить на удаленном веб-сервере. В примере программа файл-сервера расположена на компьютере разработчика в демонстрационных целях.

Рисунок 4. Порядок работы с файлами в примере

В примере работа с файлами реализована в следующем порядке:

1. Пользователь на странице тестового веб-приложения задает путь к файлу, с которым требуется выполнить одну из криптографических операций, и вызывает криптографическую операцию. При этом веб-браузер передает программе Web Unit, установленной на локальном компьютере, содержимое файла в формате Base64.
2. Программа Web Unit, взаимодействуя с криптопровайдером ViPNet CSP, выполняет криптографическую операцию и передает результат в формате Base64 в веб-браузер.
3. В ОС Windows веб-браузер передает данные, полученные от программы Web Unit, в программу тестового файл-сервера, запущенную на локальном компьютере.
4. В ОС Windows программа тестового файл-сервера преобразует данные в файл и передает ссылку на этот файл обратно в веб-браузер. При этом веб-браузер автоматически начинает скачивание файла.

В ОС Windows пример находится в следующей папке:

- `C:\Program Files\InfoTeCS\ViPNet PKI Client\SDK\Sites\WithFiles` — для 32-разрядных ОС.
- `C:\Program Files(x86)\InfoTeCS\ViPNet PKI Client\SDK\Sites\WithFiles` — для 64-разрядных ОС.

Тестовый файл-сервер находится в следующей папке:

- `C:\Program Files\InfoTeCS\ViPNet PKI Client\SDK` — для 32-разрядных ОС.
- `C:\Program Files(x86)\InfoTeCS\ViPNet PKI Client\SDK` — для 64-разрядных ОС.

В ОС Linux пример находится в каталоге `/opt/itcs/share/pki-client/Sites/WithFiles`.

Добавление вызова криптографических функций в веб-приложения

Чтобы вызывать криптографические операции, в исходный код веб-страницы необходимо добавить фрагмент на языке JavaScript. Для этого выполните следующие действия:

1 Скачайте и подключите библиотеку JavaScript jQuery версии 2.1.0 или более поздней версии):

```
<script type="text/javascript" src="Scripts/jquery-2.1.0.min.js"> </script>
```

2 Подключите библиотеку JavaScript `lss-client.js`:

```
<script src="Scripts/lss-client.js" type="text/javascript"> </script>
```

Примечание. В ОС Windows библиотека JavaScript `lss-client.js` по умолчанию находится в следующей папке:

- `C:\Program Files\InfoTeCS\ViPNet PKI Client\SDK\Sites\Base64DataOnly\Scripts` — для 32-разрядных ОС.
- `C:\Program Files (x86)\InfoTeCS\ViPNet PKI Client\SDK\Sites\Base64DataOnly\Scripts` — для 64-разрядных ОС.

В ОС Linux библиотека JavaScript `lss-client.js` по умолчанию находится в каталоге `/opt/itcs/share/pki-client/Sites/Base64/scripts`.

3 Выполните инициализацию объекта `LssClient`. Для этого выполните одно из следующих действий:

- Для вызова криптографической операции с демонстрацией графического интерфейса:

  ```
  var client = new LssClient(jQuery);
  ```

- Для вызова криптографической операции в режиме «Без уведомления» (см. Добавление вызова криптографических функций в режиме «Без подтверждения» на стр. 15):

  ```
  var client = new LssClient(jQuery).withByPass();
  ```

4 Задайте расширение, используемое в случае если пользователь не задаст расширение обрабатываемого файла на веб-странице:

```
var defaultExtension = '.doc';
```

5 Задайте интервал времени в миллисекундах, через который будет проверяться возможность соединения с программой ViPNet PKI Client Web Unit:

Примечание. Рекомендуемое значение интервала — 10000.

```
var checkLssConnectivityTimeout = 10000;
```

6 Задайте URL-адрес, по которому веб-страница будет обращаться к программе Web Unit:

`var httpBaseUrl = "http://127.0.0.1:61111/webhost";` — для подключения по протоколу HTTP;

`var sslBaseUrl = "https://127.0.0.1:61112/webhost";` — для подключения по протоколу HTTPS.

Примечание. Задаваемые URL-адреса должны совпадать с адресами, указанными в конфигурационном файле `Infotecs.PkiClientWebUnit.exe.config`, который по умолчанию находится в следующей папке:

- `C:\Program Files\Infotecs\ViPNet PKI Client\Web Unit` — для 32-разрядных ОС.
- `C:\Program Files (x86)\Infotecs\ViPNet PKI Client\Web Unit` — для 64-разрядных ОС.

7 Вызовите JavaScript-функцию `checkConnection` (см. Служебная функция checkConnection на стр. 23).

8 Вызовите JavaScript-функцию, соответствующую требуемой криптографической операции.

JavaScript-функции, используемые для вызова криптографических операций, являются методами объекта `LssClient`. В таблице ниже приведено соответствие JavaScript-функций и вызываемых ими криптографических операций.

Таблица 3. Соответствие JavaScript-функций и криптографических операций

JavaScript-функция	Криптографическая операция
`generateCertificateRequest` (см. Функция generateCertificateRequest на стр. 24)	Создание запроса на сертификат.
`installCertificateIssuedByRequest` (см. Функция installCertificateIssuedByRequest на стр. 26)	Установка сертификата, изданного по запросу, в системное хранилище.
`sign` (см. Функция sign на стр. 28)	Заверение электронной подписью.
`signFile` (см. Функция signFile на стр. 31)	Заверение электронной подписью данных объемом более 80 МБ.
`verifySign` (см. Функция verifySign на стр. 34)	Проверка электронной подписи.
`verifyFile` (см. Функция verifyFile на стр. 38)	Проверка электронной подписи данных объемом более 80 МБ.
`encrypt` (см. Функция encrypt на стр. 40)	Шифрование.
`decrypt` (см. Функция decrypt на стр. 44)	Расшифрование.

JavaScript-функция	Криптографическая операция
`signAndEncrypt` (см. Функция signAndEncrypt на стр. 47)	Заверение электронной подписью и шифрование.
`decryptAndVerifySign` (см. Функция decryptAndVerifySign на стр. 51)	Проверка электронной подписи и расшифрование.
`selectCertificate` (см. Функция specialSignFile на стр. 60)	Выбор сертификата для использования электронной подписи.
`hash` (см. Функция hash на стр. 56)	Хэширование данных. **Примечание.** В ОС Linux не поддерживается.
`hashFile` (см. Функция hashFile на стр. 57)	Хэширование данных объемом более 80 МБ. **Примечание.** В ОС Linux не поддерживается.
`specialSign01` (см. Функция specialSign01 на стр. 58)	Заверение электронной подписью со специальными параметрами. **Примечание.** В ОС Linux не поддерживается.
`specialSignFile` (см. Функция specialSignFile на стр. 60)	Заверение электронной подписью со специальными параметрами данных объемом более 80 МБ. **Примечание.** В ОС Linux не поддерживается.
`signXml` (см. Функция signXml на стр. 62)	Заверение электронной подписью данных в формате XML.
`verifySignedXml` (см. Функция verifySignedXml на стр. 67)	Проверка электронной подписи данных в формате XML.
`getCertificateList` (см. Функция getCertificateList на стр. 70)	Получение списка действительных сертификатов для заверения электронной подписью.

Служебная функция checkConnection

Функция `checkConnection` выполняет проверку возможности подключения к SDK.

Синтаксис

```
client.checkConnection()
```

Возвращаемый объект

В результате выполнения функции программа возвращает объект `Deffered`:

- В случае успешной проверки возможности подключения операция продолжается.
- В случае ошибки выдается соответствующее сообщение.

Пример

```
var client = new LssClient(jQuery);
    client.checkConnection()
    .done(function(response){
        // Программа ViPNet PKI Client SDK доступна.
    }
        .fail(function (error) {
            alert('Не удалось подключиться к ViPNet PKI Client SDK');
            console.log(error);
        })
```

Функция generateCertificateRequest

Функция `generateCertificateRequest` обращается к программе ViPNet PKI Client Web Unit для создания запроса на сертификат. Аргументом функции является объект `options`, задающий параметры запроса.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображается окно для выбора криптопровайдера, с помощью которого требуется создать контейнер ключей (см. глоссарий, стр. 81) и запрос на сертификат. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Синтаксис

```
client.generateCertificateRequest(options)
```

Входные параметры

Поля объекта `options`:

- `subject` — поля, задающие субъект сертификата.
- `extendedKeyUsages` — массив объектных идентификаторов (OID) расширений сертификата.
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `generateCertificateRequest` к программе ViPNet PKI Client Web Unit появляется окно **Создание запроса на сертификат**, в котором пользователю предлагается выбрать алгоритм открытого ключа, а также подтвердить выполнение операции.

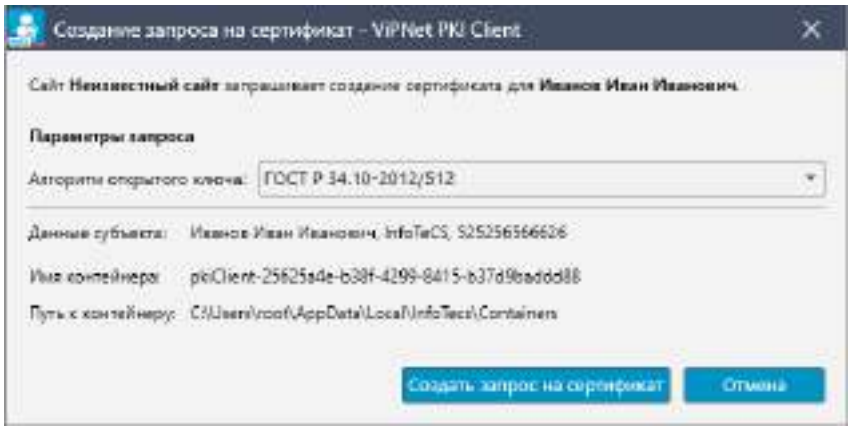

Рисунок 5. Создание запроса на сертификат (ОС Windows)

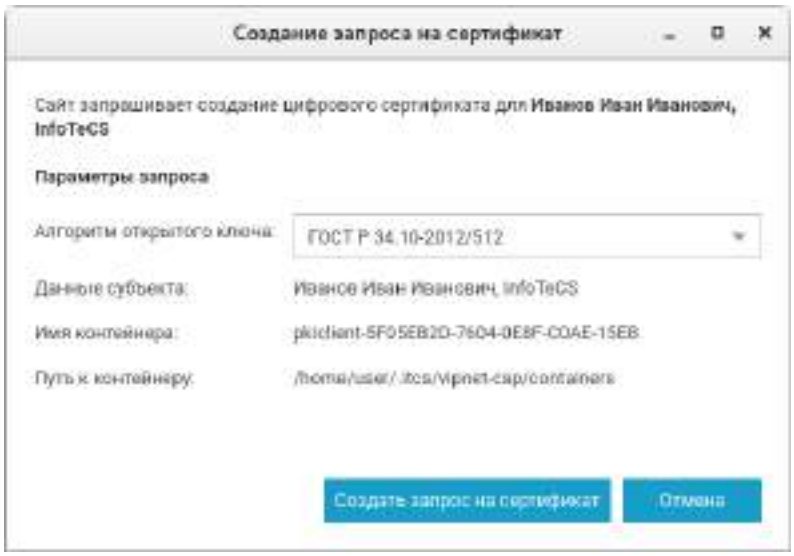

Рисунок 6. Создание запроса на сертификат (ОС Linux)

При подтверждении пользователем криптографической операции появляются окна криптопровайдера ViPNet CSP, необходимые для создания контейнера ключей.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `RawData` — запрос на сертификат в формате Base64.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
    subject: "CN=Vasiliy;SN=Petrov;INN=12345",
    extendedKeyUsages: ["2.5.29.32.0", "2.5.29.37.0"]
};
client. generateCertificateRequest (options)
.done(function (response) {
    if (response.IsSuccessful) {
        // Операция прошла успешно.
        var request = response.RawData;
    } else {
        // Операция завершилась с ошибкой.
        var error = response.ErrorMessage;
    }
})
.fail(function (error) {
    // Запрос не удалось отправить или программа
    // ViPNet PKI Client Web Unit не смогла принять подключение.
});
```

Функция installCertificateIssuedByRequest

Функция `installCertificateIssuedByRequest` обращается к программе ViPNet PKI Client Web Unit для установки сертификата, изданного удостоверяющим центром по ранее сделанному запросу (см. Функция generateCertificateRequest на стр. 24), в хранилище. Аргументом функции является объект `options`, передающий сертификат в формате Base64. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Синтаксис

```
client.installCertificateIssuedByRequest(options)
```

Входные параметры

Объект `options` имеет поле `certificate`, которое должно содержать сертификат в формате Base64.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращается объект с результатами операции либо объект с описанием ошибки.

Состав объекта при успешном выполнении операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`. Разработчик в этом случае может задать в своем веб-приложении вывод окна с уведомлением об успешном завершении операции.

Примечание. Если устанавливаемый сертификат уже есть в хранилище Windows, программа ViPNet PKI Client Web Unit также возвращает объект с флагом `IsSuccessful=true`.

- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var options = {
    certificate: “.........”,
    //сертификат в формате Base64
};
client.installCertificateIssuedByRequest(options)
    .done(function(response) {
        if (response.IsSuccessful) {
           alert('Запрос на установку сертификата обработан успешно.');
        } else {
        alert(response.ErrorMessage);
        }
    });
```

Функция sign

Функция `sign` обращается к программе ViPNet PKI Client Web Unit для заверения данных электронной подписью. Аргументом функции является объект `options`, задающий параметры электронной подписи.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления формированием электронной подписи. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Синтаксис

```
client.sign(options)
```

Входные параметры

Поля объекта `options`:

- `base64Data` — подписываемые данные в формате Base64.
- `description` — описание подписываемого документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписываемого документа.
- `fileExtension` — расширение подписываемого документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `isAttached` — флаг типа Boolean, задающий вид электронной подписи. Может принимать следующие значения:
  - `true` — прикрепленная подпись (см. глоссарий, стр. 81).
  - `false` — открепленная подпись.
- `tspServerUrl` — опциональное поле; задает адрес сервера штампов времени; если поле задано, то в электронную подпись будет добавлен штамп времени.
- `tspServerTimeout` — опциональное поле; задает тайм-аут в миллисекундах при подключении к серверу штампов времени; значение по умолчанию — 1000.
- `base64OriginalData` – исходные данные в формате Base64, которые нужно указать в случае, если к открепленной подписи требуется добавить еще одну электронную подпись.
- `base64Certificate` — сертификат, с помощью которого выполняется электронная подпись, в формате Base64. Поле используется только в режиме «Без подтверждения».
- `disableCertificateVerification` — отключает следующие параметры проверки сертификата:

- Проверка срока действия сертификата.
- Проверка целостности цепочки корневых сертификатов.
- Проверка сертификата по списку аннулированных сертификатов (CRL).
- Проверка срока действия ключа ЭП.
- Проверка назначения сертификата.

- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `sign` к программе ViPNet PKI Client Web Unit появляется окно **Подписать - ViPNet PKI Client** (ОС Windows) или **ViPNet PKI Client Web Unit** (ОС Linux), в котором пользователю предлагается выбрать сертификат для заверения документа электронной подписью и подтвердить выполнение операции.

Примечание. При подтверждении пользователем криптографической операции может появиться окно криптопровайдера, в котором запрашивается пароль к контейнеру ключей, соответствующему выбранному сертификату.

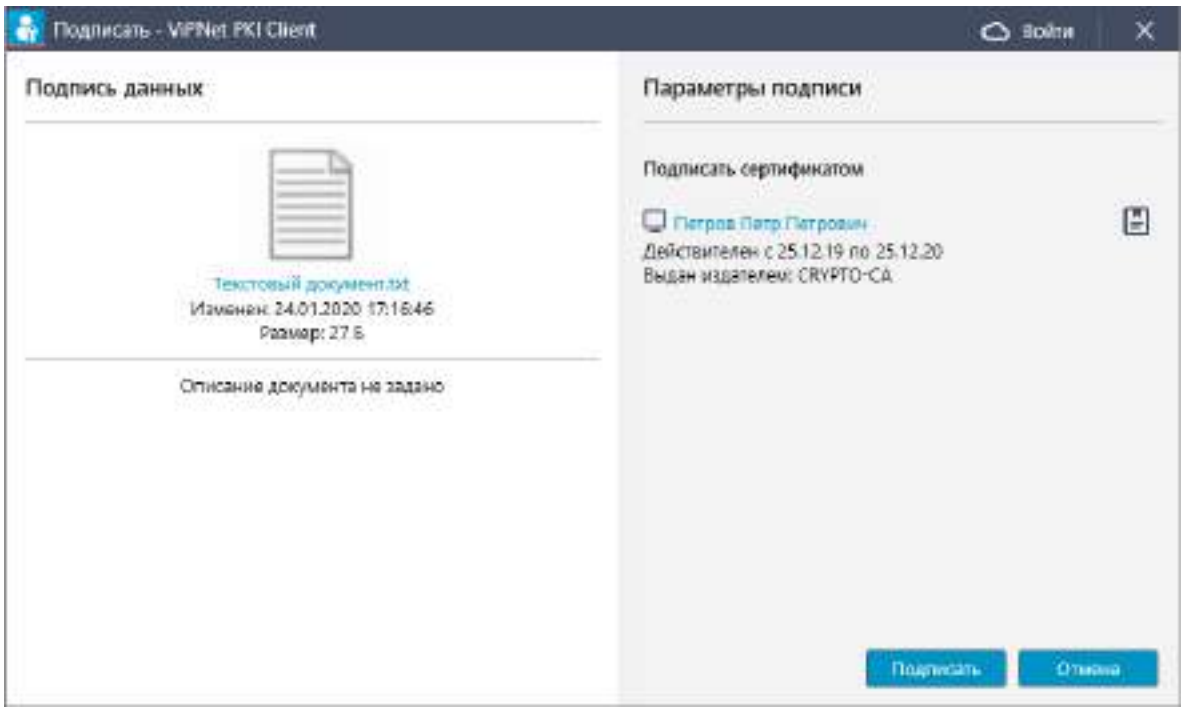

Рисунок 7. Заверение документа электронной подписью (ОС Windows)

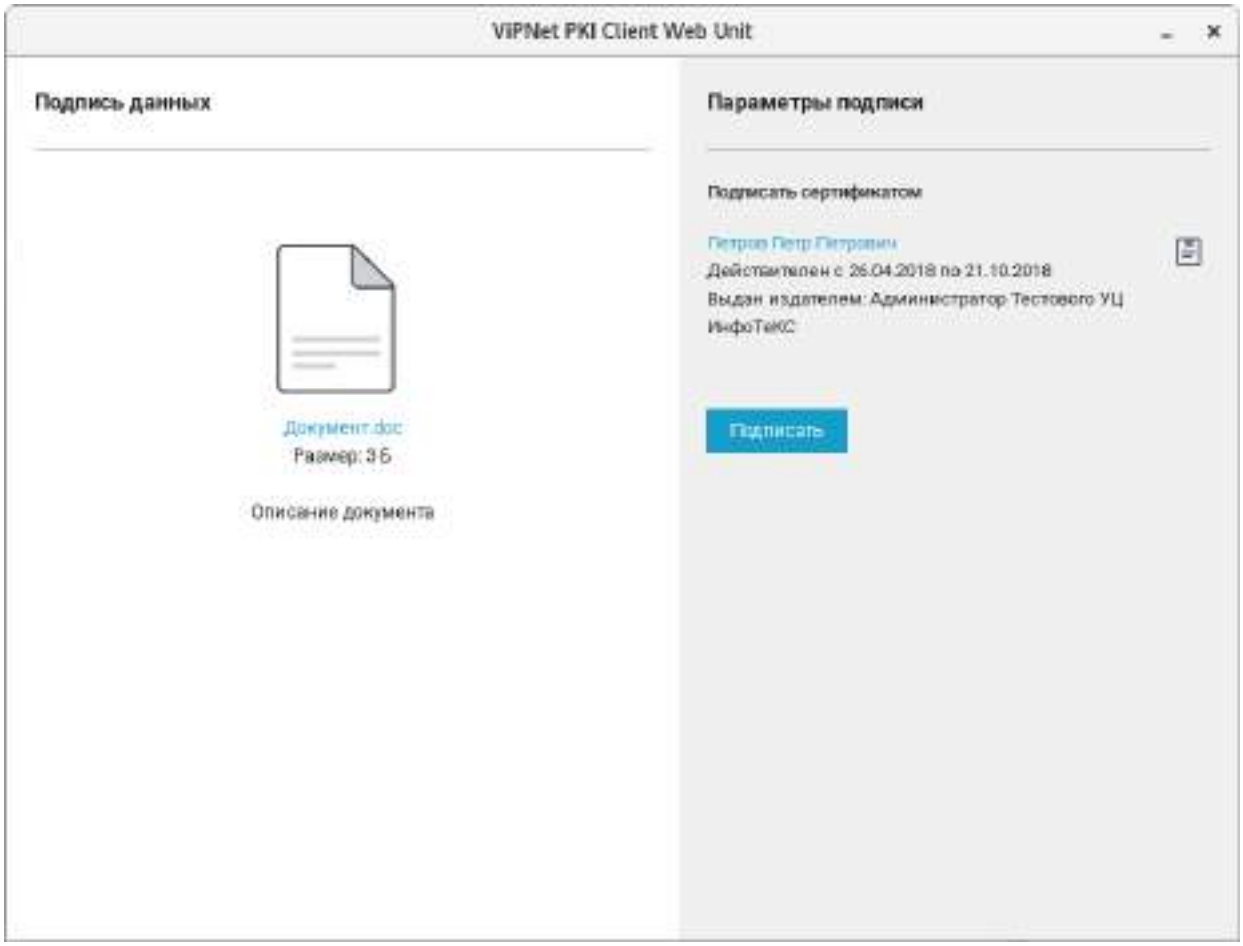

Рисунок 8. Заверение документа электронной подписью (ОС Linux)

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `SignedData` — подписанные данные в формате Base64.
- `SignCertificateJson` — сертификат, с помощью которого подписаны данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.

- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
            base64Data: "0J7RgtGH0LXRgg==",
            description: "Документ с отчетностью",
            documentName: "Отчет.doc",
            fileExtension: ".doc",
            isAttached: true
            tspServerUrl: "http://crypto:5665/tsp",
            tspServerTimeout: 1000,
        };
client.sign(options)
    .done(function (response) {
        if (response.IsSuccessful) {
            // Операция прошла успешно.
            var result = response.SignedData;
            var signCertificate = $.parseJSON(response.SignCertificateJson);
            var subject = signCertificate.Subject;
            var issuer = signCertificate.Issuer;
        } else {
            // Операция завершилась с ошибкой.
            var error = response.ErrorMessage;
        }
    })
    .fail(function (error) {
        // Запрос не удалось отправить или программа
        // ViPNet PKI Client Web Unit не смогла принять подключение.
    });
```

Функция signFile

Функция `signFile` обращается к программе ViPNet PKI Client Web Unit для заверения электронной подписью данных, объем которых превышает 80 МБ. Аргументом функции является объект `options`, задающий параметры электронной подписи.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления формированием электронной подписи. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Синтаксис

```
client.signFile(options)
```

Входные параметры

Поля объекта `options`:

- `file` — это поле содержит переменную `fileToSign`, которая является элементом `<input type="file"></input>`, содержащим выбранный для подписания файл.
- `description` — описание подписываемого документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписываемого документа.
- `fileExtension` — расширение подписываемого документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `isAttached` — флаг типа Boolean, задающий вид электронной подписи. Может принимать следующие значения:
  - `true` — прикрепленная подпись (см. глоссарий, стр. 81).
  - `false` — открепленная подпись.
- `tspServerUrl` — опциональное поле; задает адрес сервера штампов времени; если поле задано, то в электронную подпись будет добавлен штамп времени.
- `tspServerTimeout` — опциональное поле; задает тайм-аут в миллисекундах при подключении к серверу штампов времени;
- `disableCertificateVerification` — отключает следующие параметры проверки сертификата:
  - Проверка срока действия сертификата.
  - Проверка целостности цепочки корневых сертификатов.
  - Проверка сертификата по списку аннулированных сертификатов (CRL).
  - Проверка срока действия ключа ЭП.
  - Проверка назначения сертификата.
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `signFile` к программе ViPNet PKI Client Web Unit появляется окно **Подписать - ViPNet PKI Client** (см. рисунок на стр. 65) или **ViPNet PKI Client Web Unit** (см. рисунок на стр. 30), в котором пользователю предлагается выбрать сертификат для заверения документа электронной подписью и подтвердить выполнение операции.

Примечание. При подтверждении пользователем криптографической операции может появиться окно криптопровайдера, в котором запрашивается пароль к контейнеру ключей, соответствующему выбранному сертификату.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `SignatureUrl` — URL-адрес подписанных данных.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
            description: "Документ с отчетностью",
            documentName: "Отчет.doc",
            fileExtension: ".doc",
            file: fileToSign.files[0],
            isAttached: true
            tspServerUrl: "http://crypto2:8777/tsp",
            tspServerTimeout: 10000,
            disableCertificateVerification: False
};
client.signFile(options)
.done(function (response) {
        LSS.log('Получен ответ.');
        LSS.log(response);
        if (response.IsSuccessful) {
            alert('Запрос на подпись обработан успешно.');
        } else {
            alert('Запрос на подпись не обработан.');
```

```
            alert(response.ErrorMessage);
        }
});
```

Функция verifySign

Функция `verifySign` обращается к программе ViPNet PKI Client Web Unit для проверки электронной подписи. Аргументом функции является объект `options`, задающий параметры проверки электронной подписи.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления проверкой электронной подписи. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Примечание. Функция `verifySign` позволяет работать с файлами объемом не более 100 МБ.

Синтаксис

```
client.verifySign(options)
```

Входные параметры

Поля объекта `options`:

- `base64Data` — может принимать следующие значения:
  - в случае прикрепленной подписи — данные с прикрепленной подписью в формате Base64;
  - в случае откреплённой подписи — подпись в формате Base64 без подписанных данных.
- `base64DataWithoutSign` — подписанные данные в формате Base64 без электронной подписи. Поле используется только при проверке откреплённой подписи.
- `isAttached` — флаг типа Boolean, задающий вид электронной подписи. Может принимать следующие значения:
  - `true` — прикрепленная подпись.
  - `false` —откреплённая подпись.
- `description` — описание подписанного документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписанного документа.
- `fileExtension` — расширение подписанного документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `verifySign` к программе ViPNet PKI Client Web Unit появляется окно **Проверить подпись - ViPNet PKI Client** (ОС Windows) или **ViPNet PKI Client Web Unit** (ОС Linux), в котором указан результат операции.

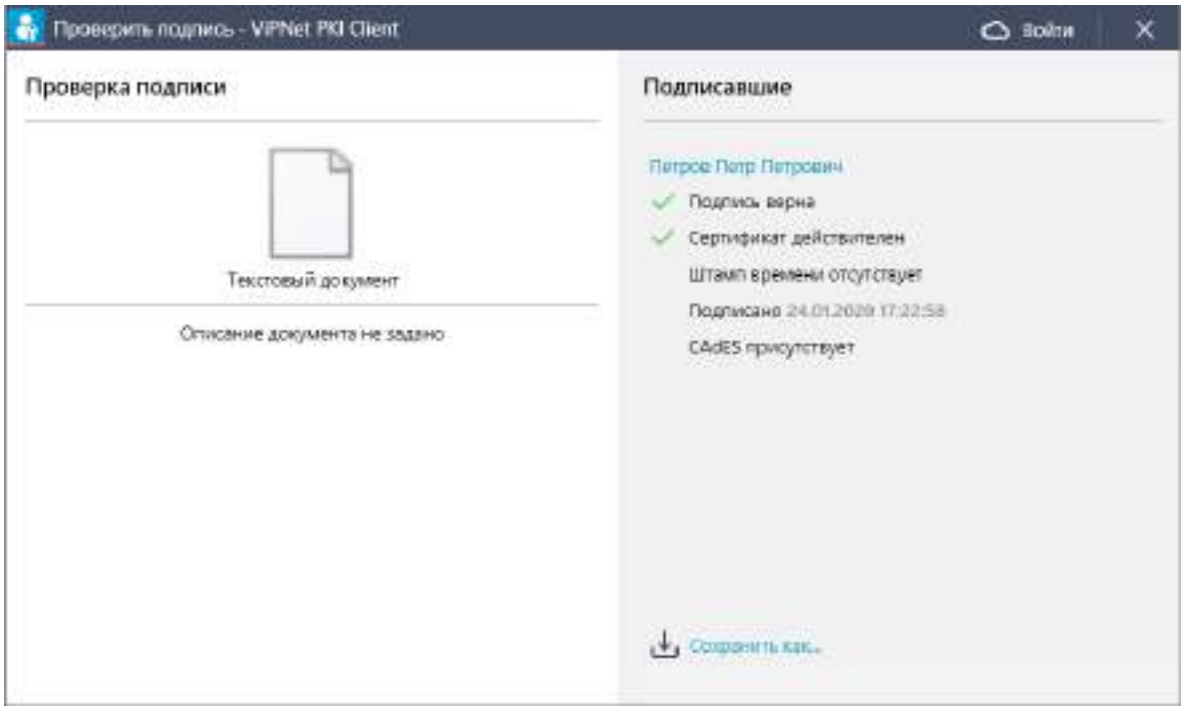

Рисунок 9. Успешная проверка электронной подписи (ОС Windows)

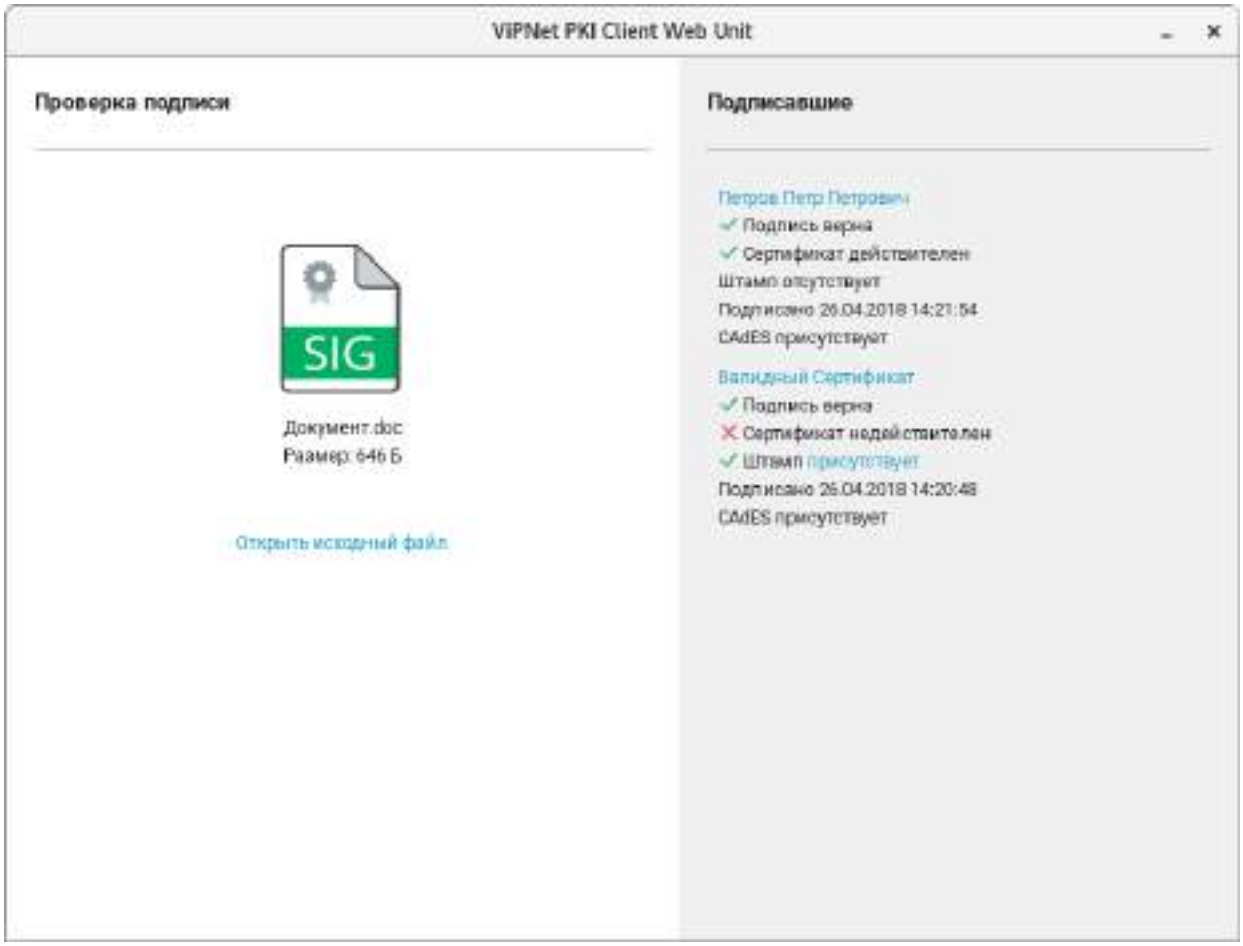

Рисунок 10. Успешная проверка электронной подписи (ОС Linux)

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `IsVerified` — флаг типа Boolean, который может принимать следующие значения:
  - `true` — все электронные подписи действительны.
  - `false` — хотя бы одна электронная подпись недействительна.
- `SignerCount` — количество электронных подписей, которыми подписан документ.
- `SignInfo` — массив объектов типа `VerifyReportItem`, содержащих информацию о результате проверки каждой электронной подписи.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта `VerifyReportItem`:

- `IsVerified` — флаг типа Boolean, принимающий следующие значения:
  - `true` — электронная подпись действительна.
  - `false` — электронная подпись недействительна.
- `SignCertificateJson` — сертификат, с помощью которого подписаны данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `CoSignInfo` — массив объектов типа `VerifyReportItem`, содержащих информацию о результате проверки каждой электронной подписи.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
    base64Data: "MIIILAYJKoZIhvcNA...",
    isAttached: true,
    description: "Документ с отчетностью",
    documentName: "Отчет.doc",
    fileExtension: ".doc",
};
client.verifySign(options)
    .done(function (response) {
        if (response.IsSuccessful) {
          // Операция прошла успешно.
          var verificationResult = response.IsVerified;
          var numberOfSigners = response.SignerCount;
          var signersInfo = response.SignInfo;
          for (var i = 0; i < signersInfo.length; i++) {
            var sign = signersInfo[i];
            var signerSubjectName = sign.SubjectName;
            var signVerified = sign.IsVerified;
            var errorMessage = sign.ErrorMessage;
            var cert = $.toJSON(sign.SignCertificateJson);
          }
        } else {
          // Операция завершилась с ошибкой.
          var error = response.ErrorMessage;
        }
    })
```

```
    .fail(function (error) {
    // Запрос не удалось отправить или программа
    // ViPNet PKI Client Web Unit не смогла принять подключение.
});
```

Функция verifyFile

Функция `verifyFile` обращается к программе ViPNet PKI Client Web Unit для проверки электронной подписи данных, объем которых превышает 80 МБ. Аргументом функции является объект `options`, задающий параметры проверки электронной подписи.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления проверкой электронной подписи. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Синтаксис

```
client.verifyFile(options)
```

Входные параметры

Поля объекта `options`:

- `file` — это поле содержит переменную, которая является элементом `<input type="file"></input>`, содержащим выбранные для проверки данные. В зависимости от типа данных может содержать следующие переменные:
  - `signatureFile` — в случае проверки данных с прикрепленной подписью;
  - `dataFile` — в случае открепленной подписи, содержит исходные данные.
- `SignatureBase64` — подпись в формате Base64 без подписанных данных.
- `isAttached` — флаг типа Boolean, задающий вид электронной подписи. Может принимать следующие значения:
  - `true` — прикрепленная подпись.
  - `false` —открепленная подпись.
- `description` — описание подписанного документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписанного документа.
- `fileExtension` — расширение подписанного документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `verifyFile` к программе ViPNet PKI Client Web Unit появляется окно **Проверить подпись - ViPNet PKI Client** (см. рисунок на стр. 68) или **ViPNet PKI Client Web Unit** (см. рисунок на стр. 36), в котором указан результат операции.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `IsVerified` — флаг типа Boolean, который может принимать следующие значения:
  - `true` — все электронные подписи действительны.
  - `false` — хотя бы одна электронная подпись недействительна.
- `SignerCount` — количество электронных подписей, которыми подписан документ.
- `SignInfo` — массив объектов типа `VerifyReportItem`, содержащих информацию о результате проверки каждой электронной подписи.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта `VerifyReportItem`:

- `IsVerified` — флаг типа Boolean, принимающий следующие значения:
  - `true` — электронная подпись действительна.
  - `false` — электронная подпись недействительна.
- `SignCertificateJson` — сертификат, с помощью которого подписаны данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `CoSignInfo` — массив объектов типа `VerifyReportItem`, содержащих информацию о результате проверки каждой электронной подписи.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `IsSuccessful` — флаг типа Boolean со значением `false`.

- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
    isAttached: true,
    description: "Документ с отчетностью",
    documentName: "Отчет.doc",
    fileExtension: ".doc",
};
if (options.isAttached) {
    options.file = signatureFile().files[0];
} else {
    options.file = dataFile().files[0];
    options.SignatureBase64 = signatureFileBase64;
};
client.verifyFile(options)
    .done(function (response) {
        LSS.log(response);
        if (response.IsSuccessful) {
          if (response.IsVerified) {
            alert('Все подписи успешно проверены.');
          } else {
            alert('Одна или несколько подписей не прошли проверку.');
          }
        } else {
          alert(response.ErrorMessage);
        }
    })
```

Функция encrypt

Функция `encrypt` обращается к программе ViPNet PKI Client Web Unit для шифрования данных. Аргументом функции является объект `options`, задающий параметры шифрования.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления шифрованием. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Примечание. Функция `encrypt` позволяет работать с файлами объемом не более 100 МБ.

Синтаксис

```
client.encrypt(options)
```

Входные параметры

Поля объекта `options`:

- `base64Data` — шифруемые данные в формате Base64.
- `base64Certificates` — массив сертификатов получателей в формате Base64.
- `description` — описание шифруемого документа, отображаемое на веб-странице.
- `documentName` — имя и расширение шифруемого документа.
- `fileExtension` — расширение шифруемого документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `disableCertificateVerification` — отключает следующие параметры проверки сертификата:
  - Проверка срока действия сертификата.
  - Проверка целостности цепочки корневых сертификатов.
  - Проверка сертификата по списку аннулированных сертификатов (CRL).
  - Проверка срока действия ключа ЭП.
  - Проверка назначения сертификата.
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `encrypt` к программе ViPNet PKI Client Web Unit появляется окно **Зашифровать - ViPNet PKI Client** (ОС Windows) или **ViPNet PKI Client Web Unit** (ОС Linux), в котором пользователю предлагается подтвердить выполнение операции.

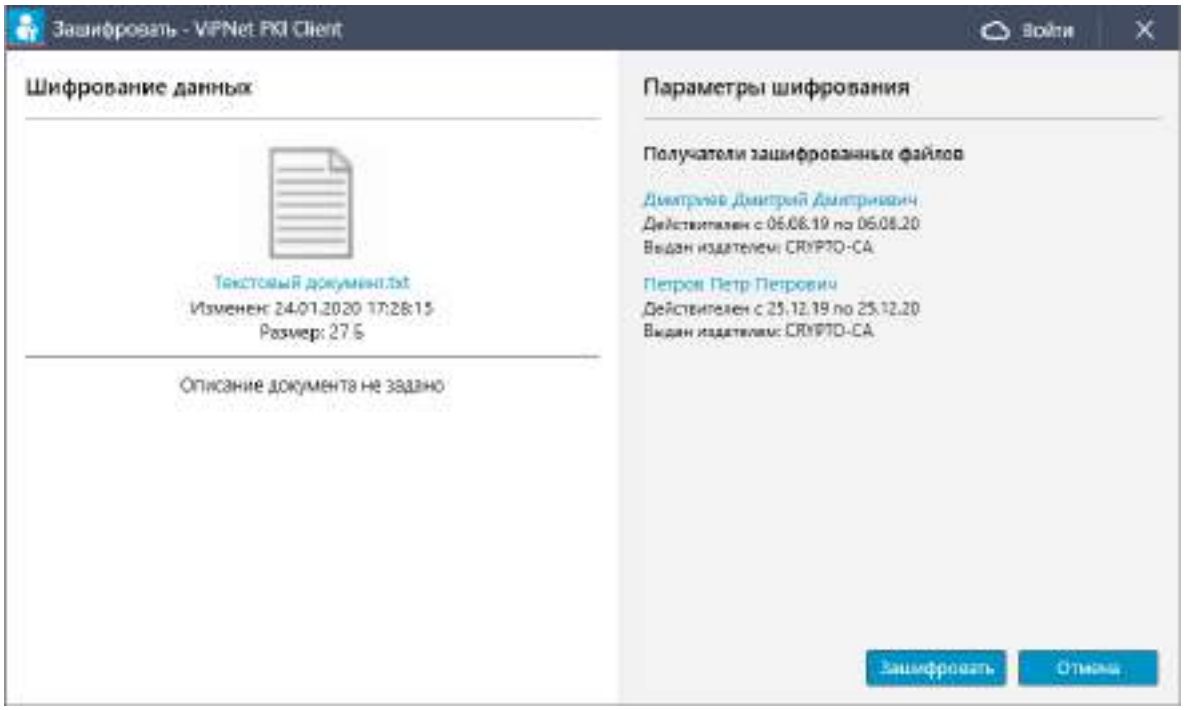

Рисунок 11. Шифрование документа (ОС Windows)

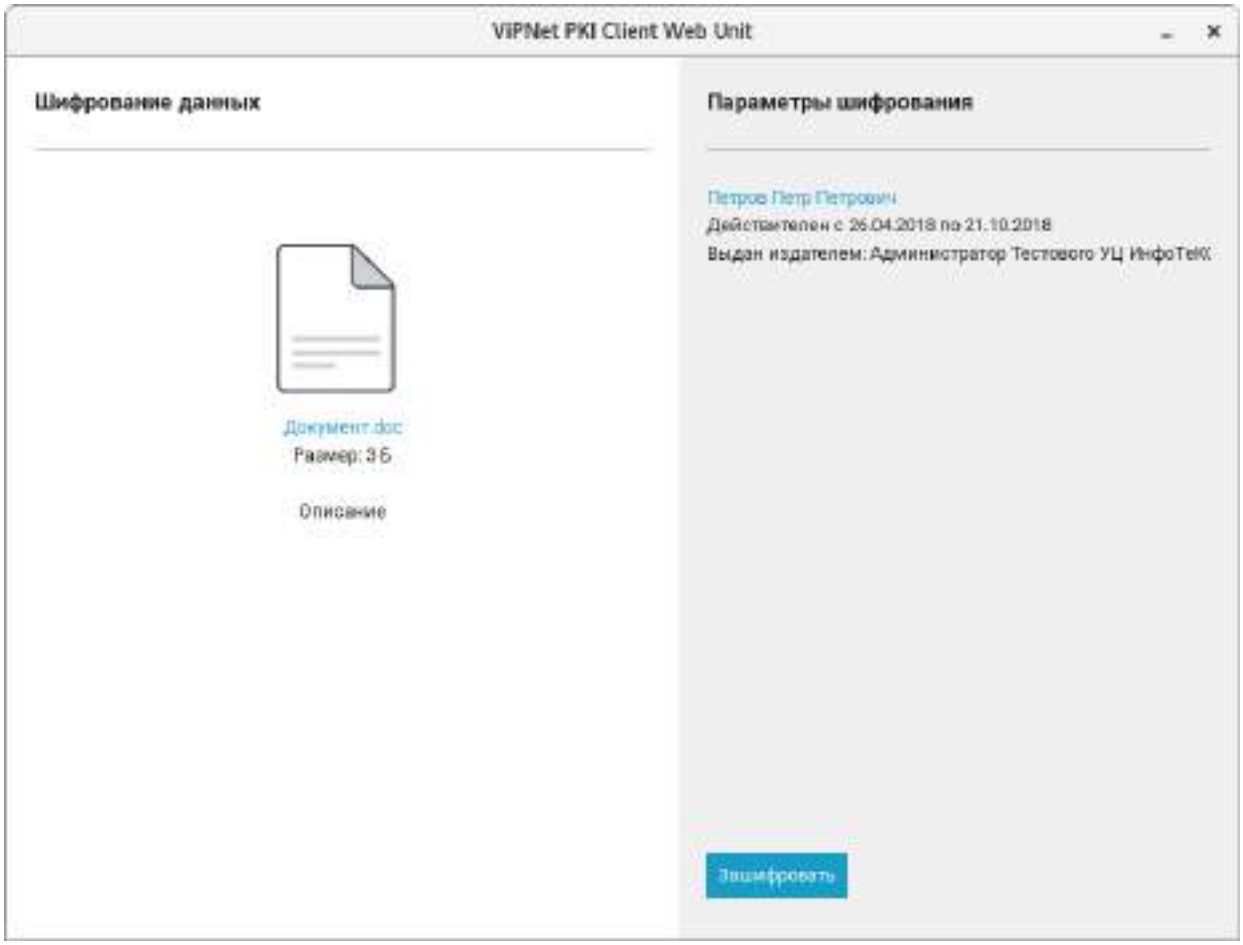

Рисунок 12. Шифрование документа (ОС Linux)

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `EncryptedData` — зашифрованные данные в формате Base64.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
    base64Data: "0J7RGTGH0LXRGG==",
    base64Certificates: ["MIIE6jCCBJ...", "MIIDDzCCAf..."],
    description: "Документ с отчетностью",
    documentName: "Отчет.doc",
    fileExtension: ".doc",
};
client.encrypt(options)
    .done(function (response) {
        if (response.IsSuccessful) {
            // Операция прошла успешно.
            var encryptedData = response.EncryptedData;
        } else {
            // Операция завершилась с ошибкой.
            var error = response.ErrorMessage;
        }
    })
    .fail(function (error) {
        // Запрос не удалось отправить или программа
        // ViPNet PKI Client Web Unit не смогла принять подключение.
    });
```

Функция decrypt

Функция `decrypt` обращается к программе ViPNet PKI Client Web Unit для расшифрования данных. Аргументом функции является объект `options`, задающий параметры расшифрования.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления расшифрованием данных. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Примечание. Функция `decrypt` позволяет работать с файлами объемом не более 100 МБ.

Синтаксис

```
client.decrypt(options)
```

Входные параметры

Поля объекта `options`:

- `base64Data` — расшифровываемые данные в формате Base64.
- `description` — описание расшифровываемого документа, отображаемое на веб-странице.
- `documentName` — имя и расширение расшифровываемого документа.
- `fileExtension` — расширение расшифровываемого документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `disableCertificateVerification` — отключает следующие параметры проверки сертификата:
  - Проверка срока действия сертификата.
  - Проверка целостности цепочки корневых сертификатов.
  - Проверка сертификата по списку аннулированных сертификатов (CRL).
  - Проверка срока действия ключа ЭП.
  - Проверка назначения сертификата.
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `decrypt` к программе ViPNet PKI Client Web Unit появляется окно **Расшифровать - ViPNet PKI Client** (ОС Windows) или **ViPNet PKI Client Web Unit** (ОС Linux), в котором пользователю предлагается подтвердить выполнение операции.

Примечание. При подтверждении пользователем криптографической операции может появиться окно криптопровайдера, в котором запрашивается пароль к контейнеру ключей, соответствующему сертификату, с помощью которого зашифрован документ.

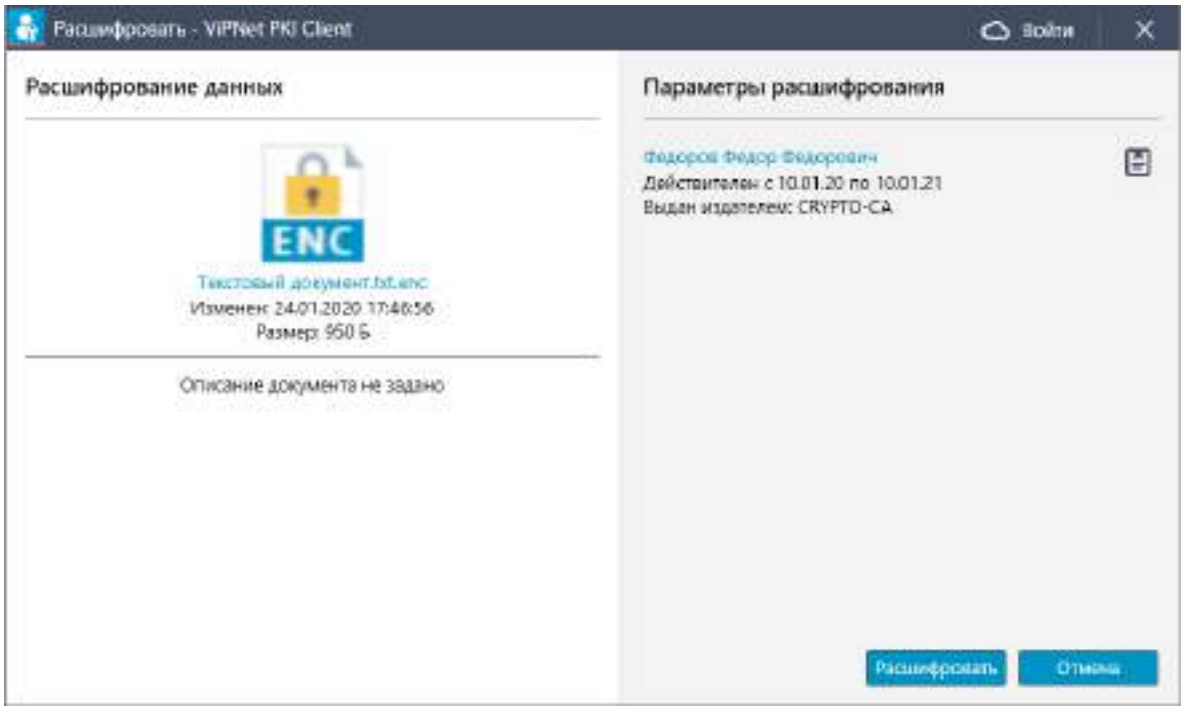

Рисунок 13. Расшифрование документа (ОС Windows)

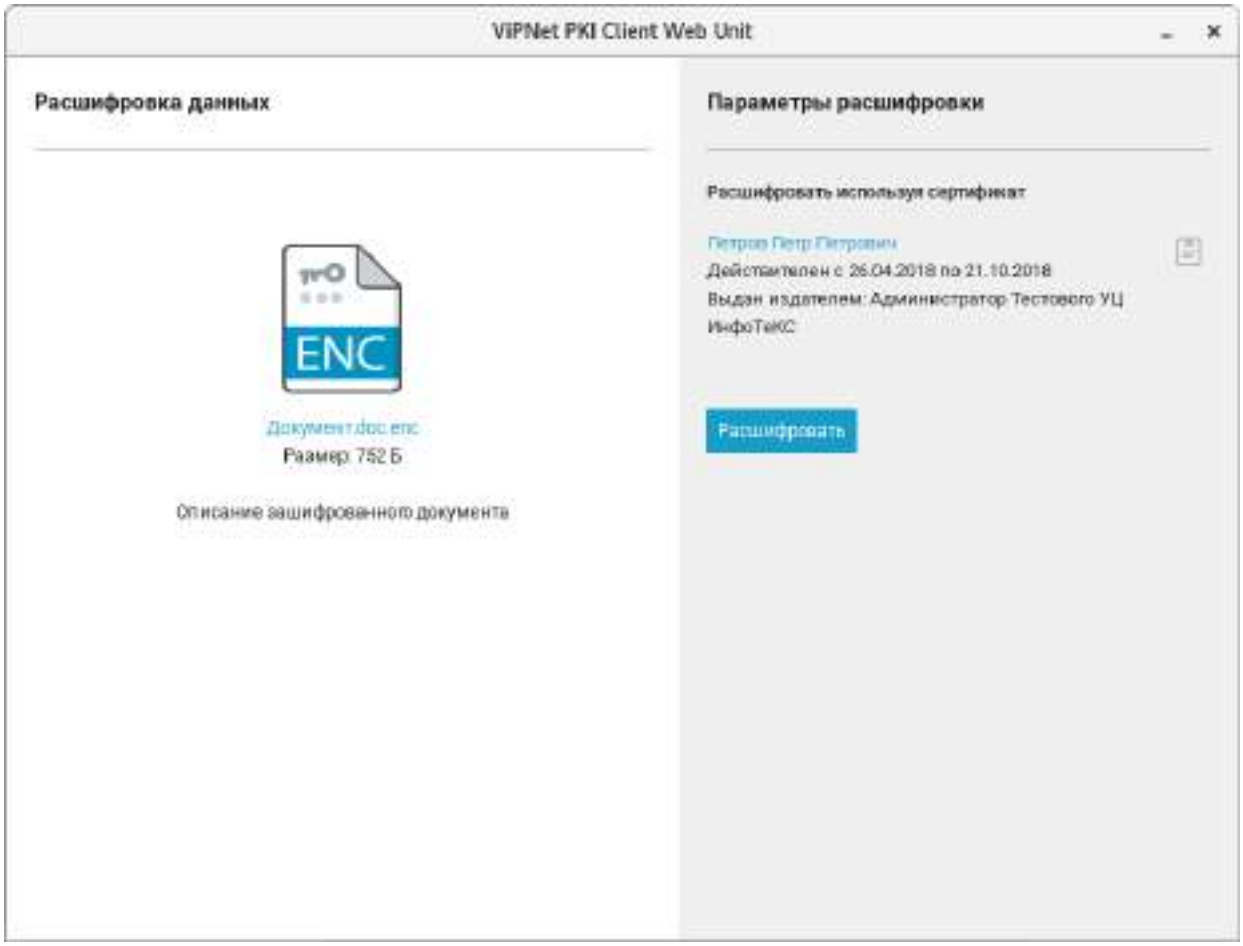

Рисунок 14. Расшифрование документа (ОС Linux)

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `DecryptionCertificateJson` — сертификат, с помощью которого расшифрованы данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70). Только для ОС Linux.
- `DecryptedData` — расшифрованные данные в формате Base64.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `IsSuccessful` — флаг типа Boolean со значением `false`.

- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
    base64Data: "MIIILAYJKo...",
    description: "Документ с отчетностью",
    documentName: "Отчет.doc",
    fileExtension: ".doc",
};
client.decrypt(options)
    .done(function (response) {
        if (response.IsSuccessful) {
            // Операция прошла успешно.
            var decryptedData = response.DecryptedData;
            var decryptionCertificate =
            $.parseJSON(response.DecryptionCertificateJson);
        } else {
            // Операция завершилась с ошибкой.
            var error = response.ErrorMessage;
        }
    })
    .fail(function (error) {
        // Запрос не удалось отправить или программа
        // ViPNet PKI Client Web Unit не смогла принять подключение.
    });
```

Функция signAndEncrypt

Функция `signAndEncrypt` обращается к программе ViPNet PKI Client Web Unit для заверения электронной подписью и шифрования данных. Аргументом функции является объект `options`, задающий параметры электронной подписи и шифрования.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления формированием электронной подписи и шифрованием. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Примечание. Функция `signAndEncrypt` позволяет работать с файлами объемом не более 100 МБ.

Синтаксис

```
client.signAndEncrypt(options)
```

Входные параметры

Поля объекта `options`:

- `base64Data` — подписываемые и шифруемые данные в формате Base64.
- `base64Certificates` — массив сертификатов получателей документа в формате Base64.
- `description` — описание подписываемого и шифруемого документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписываемого и шифруемого документа.
- `fileExtension` — расширение подписываемого и шифруемого документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `tspServerUrl` — опциональное поле; задает адрес сервера штампов времени; если поле задано, то в электронную подпись будет добавлен штамп времени.
- `tspServerTimeout` — опциональное поле; задает тайм-аут в миллисекундах при подключении к серверу штампов времени; значение по умолчанию — 1000.
- `base64SignCertificate` — сертификат, с помощью которого выполняется электронная подпись, в формате Base64. Поле используется только в режиме «Без подтверждения».
- `disableCertificateVerification` — отключает следующие параметры проверки сертификатов:
  - Проверка срока действия сертификата.
  - Проверка целостности цепочки корневых сертификатов.
  - Проверка сертификата по списку аннулированных сертификатов (CRL).
  - Проверка срока действия ключа ЭП.
  - Проверка назначения сертификата.
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `signAndEncrypt` к программе ViPNet PKI Client Web Unit появляется окно **Подписать и зашифровать - ViPNet PKI Client** (ОС Windows) или **ViPNet PKI Client Web Unit** (ОС

Linux), в котором пользователю предлагается выбрать сертификат для заверения документа электронной подписью и подтвердить выполнение операции.

Примечание. При подтверждении пользователем криптографической операции может появиться окно криптопровайдера, в котором запрашивается пароль к контейнеру ключей, соответствующему выбранному сертификату для формирования электронной подписи.

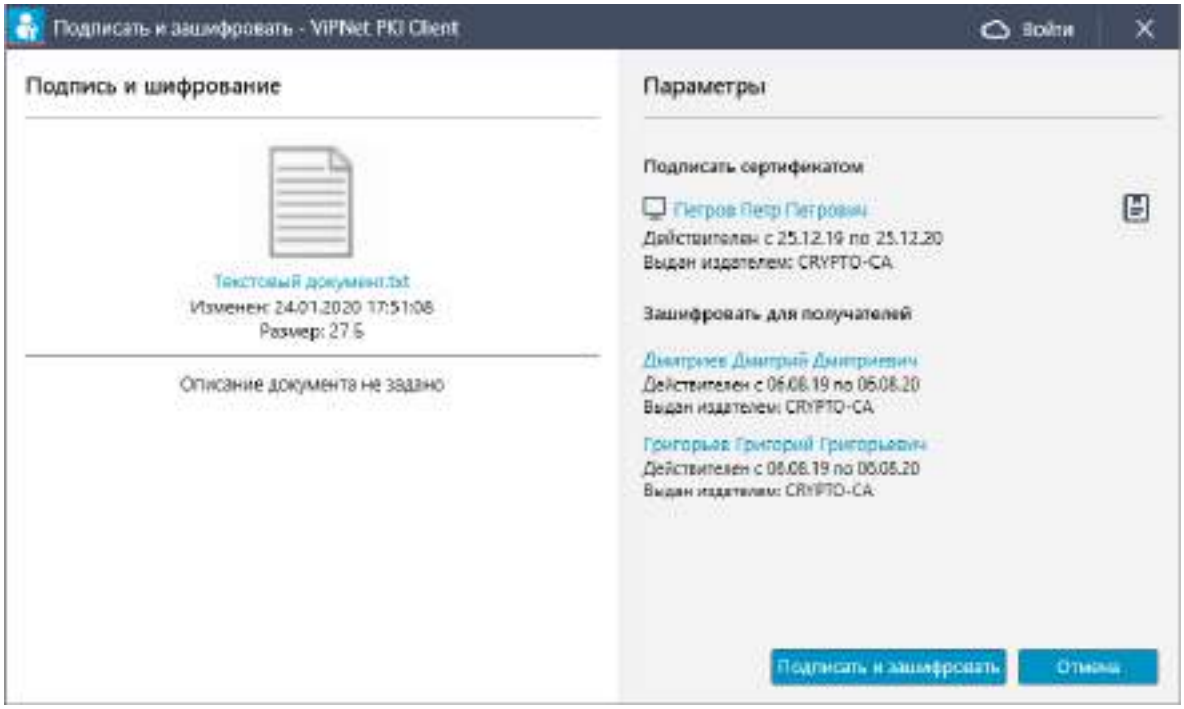

Рисунок 15. Подписание и шифрование документа (ОС Windows)

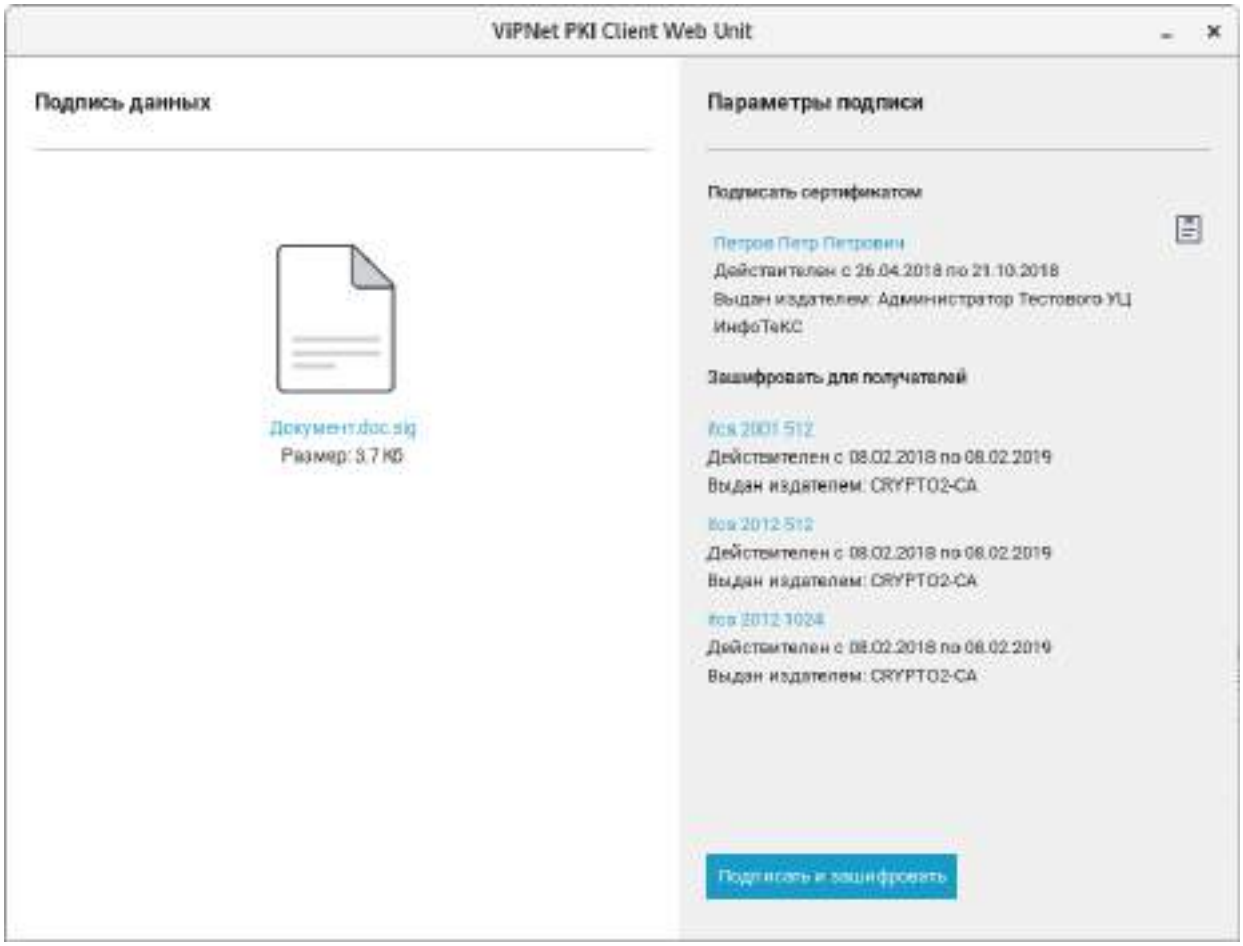

Рисунок 16. Подписание и шифрование документа (ОС Linux)

Внимание! Поддерживается подписание документа только одной электронной подписью.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `SignedAndEncryptedData` — зашифрованные и подписанные данные в формате Base64.
- `SignCertificateJson` — сертификат, с помощью которого подписаны данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
    base64Data: "0J7RgtGH0LXRgg==",
    base64Certificates: ["MIIE6jCCBJ...", "MIIDDzCCAf..."],
    description: "Документ с отчетностью",
    documentName: "Отчет.doc",
    fileExtension: ".doc",
    base64SignCertificate: "",
    tspServerUrl: "http://crypto:5665/tsp",
    tspServerTimeout: 1000,
};
client.signAndEncrypt(options)
    .done(function (response) {
        if (response.IsSuccessful) {
            // Операция прошла успешно.
            var signedAndEncryptedData = response.SignedAndEncryptedData;
            var signCertificate = $.parseJSON(response.SignCertificateJson);
        } else {
            // Операция завершилась с ошибкой.
            var error = response.ErrorMessage;
        }
    })
    .fail(function (error) {
        // Запрос не удалось отправить или программа
        // ViPNet PKI Client Web Unit не смогла принять подключение.
    });
```

Функция decryptAndVerifySign

Функция `decryptAndVerifySign` обращается к программе ViPNet PKI Client Web Unit для расшифрования и проверки электронной подписи данных. Аргументом функции является объект `options`, задающий параметры проверки электронной подписи и расшифрования.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления проверкой электронной подписи и

расшифрованием данных. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Примечание. Функция `decryptAndVerifySign` позволяет работать с файлами объемом не более 100 МБ.

Синтаксис

```
client.decryptAndVerifySign(options)
```

Входные параметры

Поля объекта `options`:

- `base64Data` — подписанные и зашифрованные данные в формате Base64.
- `description` — описание подписанного и зашифрованного документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписанного и зашифрованного документа.
- `fileExtension` — расширение подписанного и зашифрованного документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `disableCertificateVerification` — отключает следующие параметры проверки сертификата, с помощью которого будет расшифрованы данные:
  - Проверка срока действия сертификата.
  - Проверка целостности цепочки корневых сертификатов.
  - Проверка сертификата по списку аннулированных сертификатов (CRL).
  - Проверка срока действия ключа ЭП.
  - Проверка назначения сертификата.
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `decryptAndVerifySign` к программе ViPNet PKI Client Web Unit появляется окно, в котором пользователю предлагается подтвердить операцию расшифрования:

- ОС Windows — **Расшифровать - ViPNet PKI Client** (см. рисунок на стр. 45).
- ОС Linux — **ViPNet PKI Client Web Unit** (см. рисунок на стр. 46).

А затем окно с результатом проверки подписи:

- ОС Windows — **Проверить подпись - ViPNet PKI Client** (см. рисунок на стр. 68).
- ОС Linux — **ViPNet PKI Client Web Unit** (см. рисунок на стр. 36).

Примечание. При подтверждении пользователем операции расшифрования может появиться окно криптопровайдера с запросом ввести пароль к контейнеру ключей, соответствующему сертификату, с помощью которого зашифрован документ.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `OriginalData` — расшифрованные данные в формате Base64 с действительной электронной подписью.
- `VerifyReport` — объект, аналогичный объекту, возвращаемому программой при выполнении функции `verifySign` (см. Функция verifySign на стр. 34).
- `DecryptionCertificateJson` — сертификат, с помощью которого расшифрованы данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70). Только для ОС Linux.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
    base64Data: "MIIILAYJKo...",
    description: "Документ с отчетностью",
    documentName: "Отчет.doc",
    fileExtension: ".doc",
};
```

```
client.decryptAndVerifySign(options)
    .done(function(response) {
        if (response.IsSuccessful) {
          // Операция прошла успешно.
          var originalData = response.OriginalData;
          var decryptionCertificate =
          $.parseJSON(response.DecryptionCertificateJson);
          var verifyReport = response.VerifyReport;
          var verificationResult = verifyReport.IsVerified;
          var numberOfSigners = verifyReport.SignerCount;
          var signersInfo = verifyReport.SignInfo;
          for (var i = 0; i < signersInfo.length; i++) {
            var sign = signersInfo[i];
            var signerSubjectName = sign.SubjectName;
            var signVerified = sign.IsVerified;
            var errorMessage = sign.ErrorMessage;
            var cert = $.toJSON(sign.SignCertificateJson);
        } else {
            // Операция завершилась с ошибкой.
            var error = response.ErrorMessage;
        }
    })
    .fail(function (error) {
        // Запрос не удалось отправить или программа
        // ViPNet PKI Client Web Unit не смогла принять подключение.
        });
```

Функция selectCertificate

Функция `selectCertificate` позволяет пользователю веб-приложения выбрать сертификат, с помощью которого будет выполняться электронная подпись. Аргументом функции является необязательный объект `options`, позволяющий отключить следующие параметры проверки сертификатов:

- Проверка срока действия сертификата.
- Проверка целостности цепочки корневых сертификатов.
- Проверка сертификата по списку аннулированных сертификатов (CRL).
- Проверка срока действия ключа ЭП.
- Проверка назначения сертификата.

При отключении указанных параметров проверки в окне выбора сертификата будут отображены также сертификаты, которые не прошли бы проверку по данным параметрам.

Синтаксис

```
client.selectCertificate(options)
```

Входные параметры

Поле объекта `options`:

- `disableCertificateVerification` — отключает указанные выше параметры проверки сертификатов.

Возвращаемый объект

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `ErrorCode` — константа со значением `0`.
- `TextFormat` — строковое представление сертификата.
- `CertDetails` — подробное представление сертификата.
- `CertificateJson` — сертификат в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `CertificateJson` — сертификат в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
client.selectCertificate()
    .done(function (response) {
        if (response.IsSuccessful) {
            // Операция прошла успешно.
            var cert = $.parseJSON(response.CertificateJson);
            var issuer = cert.Issuer;
            var subject = cert.Subject;
            var serialNumber = cert.SerialNumber;
            var rawData = cert.Base64RawData;
            var notBefore = new Date(parseInt(cert.NotBefore.substr(6)));
            var notAfter = new Date(parseInt(cert.NotAfter.substr(6)));
        } else {
```

```
            // Операция завершилась с ошибкой.
            var error = response.ErrorMessage;
        }
    })
    .fail(function(error) {
        // Запрос не удалось отправить или программа
        // ViPNet PKI Client Web Unit не смогла принять подключение.
    });
```

Функция hash

Функция `hash` обращается к программе ViPNet PKI Client Web Unit для заверения хэширования данных. Аргументом функции является объект `options`, в котором передаются хэшируемые данные.

Примечание. Функция `hash` позволяет работать с файлами объемом не более 100 МБ.

Синтаксис

```
client.hash(options)
```

Входные параметры

Поля объекта `options`:

- `base64Data` — данные в формате Base64, которые необходимо хэшировать
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `Hash` — хэшированные данные в формате Base64.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.

- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var options = {
    base64Data: base64Data,
};
    client.hash(options)
    .done(function(response) {
      if (response.IsSuccessful) {
        var hash = response.Hash;// хэш в кодировке base64
      } else {
        alert(response.ErrorMessage);
      }
});
```

Функция hashFile

Функция `hashFile` обращается к программе ViPNet PKI Client Web Unit для заверения хэширования данных, объем которых превышает 80 МБ. Аргументом функции является объект `options`, в котором передаются хэшируемые данные.

Синтаксис

```
client.hashFile(options)
```

Входные параметры

Поля объекта `options`:

- `file` — в этом поле указана переменная `fileToHash`, которая является элементом `<input type="file"></input>`, содержащим данные для хэширования.
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `Hash` — хэшированные данные.

- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery)
var options = {
    file: fileToHash.files[0],
};
client.hashFile(options)
  .done(function(response) {
    if (response.IsSuccessful) {
        alert(response.Hash);
    } else {
        alert(response.ErrorMessage);
    }
});
```

Функция specialSign01

Примечание. Функция `specialSign01` предназначена для использования в информационной системе конкретного заказчика. Если вам не поступало дополнительных указаний, для формирования электронной подписи используйте функцию `sign`.

Функция `specialSign01` может использоваться только в режиме «Без подтверждения» (см. Добавление вызова криптографических функций в режиме «Без подтверждения» на стр. 15).

Функция `specialSign01` обращается к программе ViPNet PKI Client Web Unit для заверения данных электронной подписью, но при этом программа возвращает электронную подпись в чистом виде (не в форме CMS-сообщения), а также хэш-сумму подписываемых данных. Аргументом функции является объект `options`, задающий параметры электронной подписи.

Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Примечание. Функция `specialSign01` позволяет работать с файлами объемом не более 100 МБ.

Синтаксис

```
client.specialSign01(options)
```

Входные параметры

Поля объекта `options`:

- `base64Data` — подписываемые данные в формате Base64.
- `base64Certificate` — сертификат, с помощью которого выполняется электронная подпись, в формате Base64. Поле используется только в режиме «Без подтверждения».

Примечание. В отличие от функции `sign` отключить проверку сертификата по списку CRL и проверку целостности цепочки сертификации нельзя.

- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `ErrorCode` — константа со значением `0`.
- `Hash` — хэш-сумма подписываемых данных.
- `Sign` — сформированная электронная подпись данных в формате Base64 (не в форме CMS-сообщения).
- `SignCertificateJson` — сертификат, с помощью которого подписаны данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
var options = {
            base64Data: "0J7RgtGH0LXRgg==",
            base64Certificate: "MIIDCTCCAragAwIBAgIQAdDEi2m...",
        };
client.sign(options)
    .done(function (response) {
        if (response.IsSuccessful) {
            // Операция прошла успешно.
            var hash = response.Hash;
            var sign = response.Sign;
        } else {
            // Операция завершилась с ошибкой.
            var error = response.ErrorMessage;
        }
    })
    .fail(function (error) {
        // Запрос не удалось отправить или программа
        // ViPNet PKI Client Web Unit не смогла принять подключение.
    });
```

Функция specialSignFile

Примечание. Функция `specialSignFile` предназначена для использования в информационной системе конкретного заказчика. Если вам не поступало дополнительных указаний, для формирования электронной подписи используйте функцию `sign`.

Функция `specialSignFile` обращается к программе ViPNet PKI Client Web Unit для заверения данных объемом свыше 80 МБ электронной подписью. При этом программа возвращает электронную подпись в чистом виде (не в форме CMS-сообщения), а также хэш-сумму подписываемых данных. Аргументом функции является объект `options`, задающий параметры электронной подписи.

Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Синтаксис

```
client.specialSignFile(options)
```

Входные параметры

Поля объекта `options`:

- `description` — описание подписанного документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписанного документа.

- `fileExtension` — расширение подписанного документа.
- `disableCertificateVerification` — отключает следующие параметры проверки сертификата.
  - Проверка срока действия сертификата.
  - Проверка целостности цепочки корневых сертификатов.
  - Проверка сертификата по списку аннулированных сертификатов (CRL).
  - Проверка срока действия ключа ЭП.
  - Проверка назначения сертификата.
- `file` — это поле содержит переменную `fileToSign`, которая является элементом `<input type="file"></input>`, содержащим выбранный для подписания файл.
- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `Hash` — хэш-сумма подписываемых данных.
- `Sign` — сформированная электронная подпись данных в формате Base64 (не в форме CMS-сообщения).
- `SignCertificateJson` — сертификат, с помощью которого подписаны данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var client = new LssClient(jQuery);
```

```
var options = {
    description: "Документ с отчетностью",
    documentName: "Отчет.doc",
    fileExtension: ".doc",
    disableCertificateVerification: False, //Отключение проверки сертификата
    file: fileToSign.files[0]
};
client.specialSignFile(options)
    .done(function(response) {
          if (response.IsSuccessful) {
            alert('Подпись выполнена.');
          } else {
            alert('Запрос на подпись не обработан.');
          }
});
```

Функция signXml

Функция `signXml` обращается к программе ViPNet PKI Client Web Unit для заверения данных в формате XML электронной подписью. Аргументом функции является объект `options`, задающий параметры электронной подписи.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса отображаются окна, необходимые для настройки и управления формированием электронной подписи. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Примечание. Функция `signXml` позволяет работать с файлами объемом не более 100 МБ.

Внимание! В текущей версии программы ViPNet PKI Client Web Unit не предусмотрена поддержка следующих стандартов и объектов:

- Стандарты электронной подписи Detach и Enveloping.
- Трансформация XSLT.
- Определяющий параметры подписи объект `signatureProperties`.

Синтаксис

```
client.signXml(options)
```

Входные параметры

Поля объекта `options`:

- `xml` — подписываемые данные в формате XML.

- `canonicalizationMethod` — опциональное поле; задает один из следующих алгоритмов каноникализации:
  - C14N (http://www.w3.org/TR/2001/REC-xml-c14n-2001031514),
  - C14NC (http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments),
  - EXSLUSIVE_C14N (http://www.w3.org/2001/10/xml-exc-c14n#) (выбран по умолчанию),
  - EXSLUSIVE_C14NC (http://www.w3.org/2001/10/xml-exc-c14n#WithComments),
  - C14N11 (http://www.w3.org/2006/12/xml-c14n11),
  - C14NC11 (http://www.w3.org/2006/12/xml-c14n11#WithComments).

Примечание. Эти значения находятся в объекте `JavaScript LssConstants.XmlCanonicalization`.

- `signatureLocationPath` — опциональное поле; задает значение атрибута `XPath`, который определяет место подписи в документе. По умолчанию подпись будет встроена в корневой тег.
- `signatureId` — опциональное поле; атрибут `ID` элемента `Signature`, позволяющий идентифицировать подпись, например, в случае ее поиска. По умолчанию имеет значение `null` (будет сгенерирован автоматически).
- `references` — опциональное поле; задает массив ссылок, указывающих на фрагменты документа, которые следует подписать.

Примечание. По умолчанию подписывается весь документ с трансформацией `ENVELOPED`.

Элементами массива являются следующие объекты:

- `id` — идентификационный номер элемента `Reference`.
- `uri` — задает ссылку на подписываемый трансформированный фрагмент документа, например, `#Object`. Чтобы задать ссылку на весь документ, используйте пустую строку `""`.
- `type` — задает URL-адрес, указывающий на подписываемые трансформированные данные. Например, `Type="http://www.w3.org/2000/09/xmldsig#Object"`.
- `transforms` — массив объектов, позволяющих выполнить трансформацию выбранных фрагментов XML-данных. Например, можно задать алгоритм фильтрации данных, которые следует подписать. Объект трансформации имеет следующие поля:
  - `algorithm` — задает один из следующих алгоритмов трансформации исходных данных: XPATH,
    BASE64,
    ENVELOPED,
    XPATH_FILTER2.

Примечание. Эти значения находятся в объекте `JavaScript LssConstants.XmlTransformation`.

В качестве алгоритма вы также можете задать один из методов каноникализации.

- `transformValue` — задает дополнительные данные в текстовом формате, если они предполагаются выбранным алгоритмом.

- `base64Certificate` — сертификат, с помощью которого выполняется электронная подпись, в формате Base64. Поле используется только в режиме «Без подтверждения».
- `description` — описание подписываемого документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписываемого документа. По умолчанию задано значение `Документ Xml.xml`.
- `fileExtension` — расширение подписываемого документа. По умолчанию задано значение `.xml`.
- `signatureMethod` — алгоритм подписи; вычисляется автоматически по заданному сертификату.
- `digestMethod` — алгоритм хэширования; вычисляется автоматически по заданному сертификату.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `disableCertificateVerification` — отключает следующие параметры проверки сертификата:
  - Проверка срока действия сертификата.
  - Проверка целостности цепочки корневых сертификатов.
  - Проверка сертификата по списку аннулированных сертификатов (CRL).
  - Проверка срока действия ключа ЭП.
  - Проверка назначения сертификата.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `signXml` к программе ViPNet PKI Client Web Unit появляется окно **Подписать - ViPNet PKI Client**, в котором пользователю предлагается выбрать сертификат для заверения документа электронной подписью и подтвердить выполнение операции.

Примечание. При подтверждении пользователем криптографической операции может появиться окно криптопровайдера, в котором запрашивается пароль к контейнеру ключей, соответствующему выбранному сертификату.

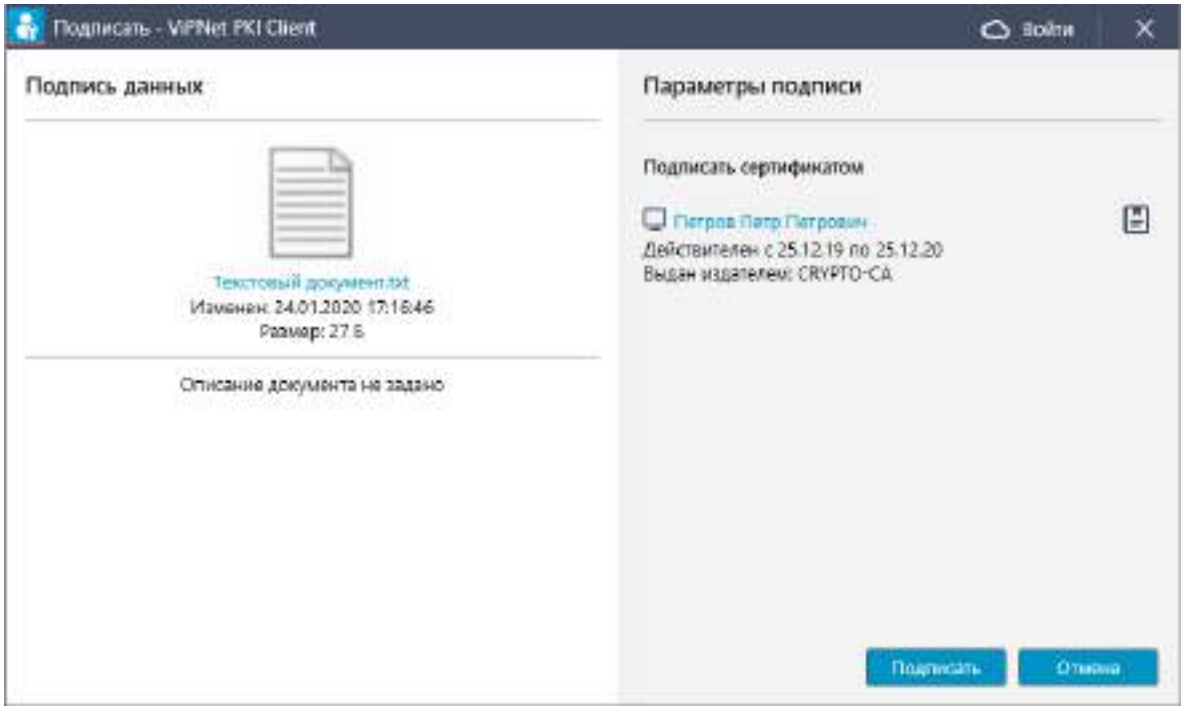

Рисунок 17. Заверение документа электронной подписью (ОС Windows)

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `SignedXml` — подписанные данные в формате XML.
- `SignCertificateJson` — сертификат, с помощью которого подписаны данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример параметров электронной подписи по умолчанию

```
var options = {
    xml: '<declaration>...</declaration>',
      documentName: 'Налоговая декларация.xml',
};
client.signXml(options)
    .done(function(response) {
      if (response.IsSuccessful) {
        //успех
      } else {
        //ошибка
        alert(response.ErrorMessage);
      }
    });
```

Пример параметров электронной подписи с трансформацией XPATH

```
var options = {
    xml: '<declaration>...</declaration>',
    documentName: 'Налоговая декларация.xml',
    references: [{
          transforms: [
    {
            algorithm: LssConstants.XmlTransformation.XPATH,
            transformValue: '<ds:Transform
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
            Algorithm="http://www.w3.org/TR/1999/REC-xpath-19991116"><ds:XPath>not(
            ancestor-or-self::ds:Signature)</ds:XPath></ds:Transform>'
          },
    ]
        }]
};
client.signXml(options)
    .done(function(response) {
      if (response.IsSuccessful) {
        //успех
      } else {
        //ошибка
        alert(response.ErrorMessage);
      }
    });
```

Пример параметров подписи с трансформацией XPATH_FILTER2

```
var options = {
    xml: '<declaration>...</declaration>',
    documentName: 'Налоговая декларация.xml',
    references: [{
          transforms: [
    {
            algorithm: LssConstants.XmlTransformation.XPATH_FILTER2,
```

```
            transformValue: '<ds:Transform
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
            Algorithm="http://www.w3.org/2002/06/xmldsig-filter2"><XPath
            Filter="subtract" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
            xmlns="http://www.w3.org/2002/06/xmldsig-filter2">here()/ancestor::ds:S
            ignature[1]</XPath></ds:Transform>'
          },
    ]
        }]
};
client.signXml(options)
    .done(function(response) {
      if (response.IsSuccessful) {
        //успех
      } else {
        //ошибка
        alert(response.ErrorMessage);
      }
    });
```

Функция verifySignedXml

Функция `verifySignedXml` обращается к программе ViPNet PKI Client Web Unit для проверки электронной подписи данных в формате XML. Аргументом функции является объект `options`, задающий параметры проверки электронной подписи.

При вызове программы ViPNet PKI Client Web Unit с демонстрацией графического интерфейса (см. Добавление вызова криптографических функций в веб-приложения на стр. 21) отображаются окна, необходимые для настройки и управления формированием электронной подписи. Результат операции программа ViPNet PKI Client Web Unit передает в веб-браузер.

Синтаксис

```
client.verifySignedXml(options)
```

Входные параметры

Поля объекта `options`:

- `SignXml` — подписанные данные в формате XML.
- `description` — описание подписанного документа, отображаемое на веб-странице.
- `documentName` — имя и расширение подписанного документа.
- `fileExtension` — расширение подписанного документа.

Примечание. Поле `fileExtension` применяется для того, чтобы пользователь смог просмотреть содержимое документа с помощью программы (например, текстового редактора), установленной на компьютере.

- `requestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пользовательский интерфейс программы ViPNet PKI Client Web Unit при обращении к ней функции

При обращении функции `verifySignXml` к программе ViPNet PKI Client Web Unit появляется окно **Проверить подпись - ViPNet PKI Client**, в котором указан результат операции.

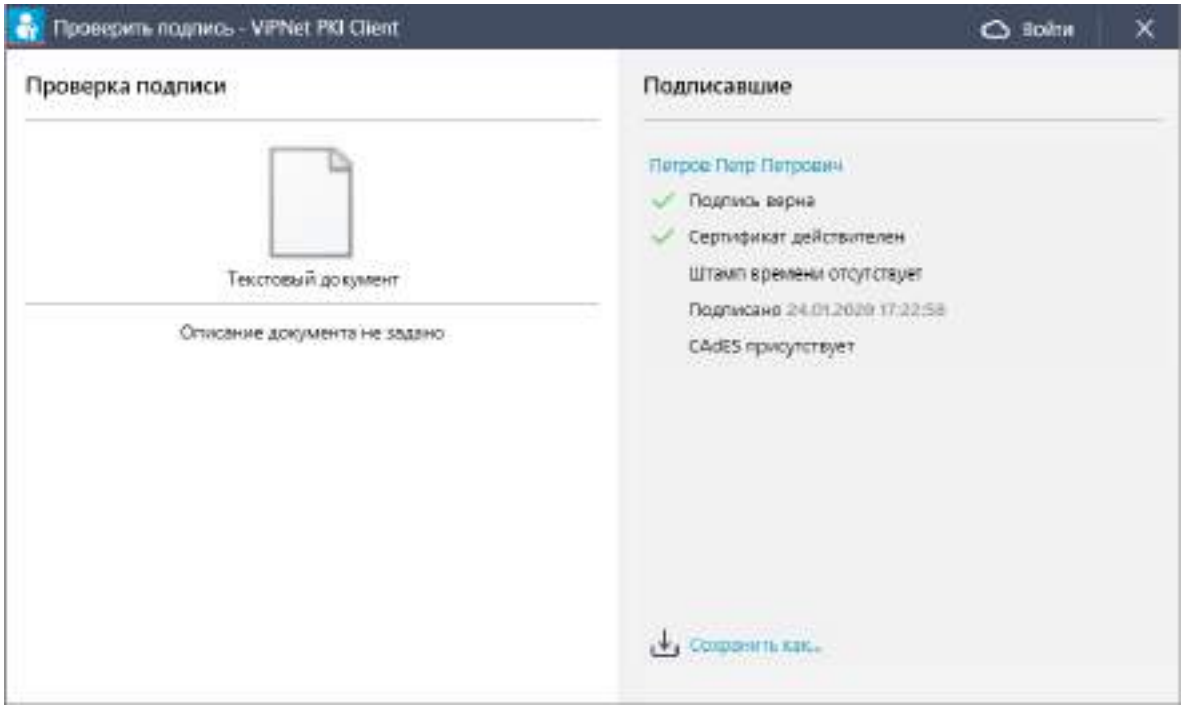

Рисунок 18. Успешная проверка электронной подписи (ОС Windows)

Объект, возвращаемый программой ViPNet PKI Client Web Unit

В результате выполнения функции программа ViPNet PKI Client Web Unit возвращает объект с результатами операции либо объект с описанием ошибки.

Состав объекта с результатами операции:

- `ErrorCode` — константа со значением `0`.
- `IsSuccessful` — флаг типа Boolean со значением `true`.
- `IsVerified` — флаг типа Boolean, который может принимать следующие значения:
  - `true` — все электронные подписи действительны.
  - `false` — хотя бы одна электронная подпись недействительна.

- `SignerCount` — количество электронных подписей, которыми подписан документ.
- `SignInfo` — массив объектов типа `VerifyReportItem`, содержащих информацию о результате проверки каждой электронной подписи.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта `VerifyReportItem`:

- `ErrorCode` (см. Коды возврата ошибок на стр. 71) — константа с кодом ошибки.
- IsVerified — флаг типа Boolean, принимающий следующие значения:
  - `true` — электронная подпись действительна.
  - `false` — электронная подпись недействительна.
- `SignCertificateJson` — сертификат, с помощью которого подписаны данные, в формате JSON (см. Сертификаты в формате JSON на стр. 70).
- `CoSignInfo` — массив объектов типа `VerifyReportItem`, содержащих информацию о результате проверки каждой электронной подписи.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Состав объекта с описанием ошибки:

- `IsSuccessful` — флаг типа Boolean со значением `false`.
- `ErrorMessage` — текстовое описание ошибки.
- `RequestId` — идентификатор запроса, задается произвольно. Служит для сопоставления запросов к программе и ответов, полученных от программы.

Пример

```
var options = {
  signedXml: <xmlWithSignatures>...</xmlWithSignatures>,
  documentName: 'Подписанный Xml',
};
client.verifySignedXml(options)
      .done(function(response) {
        if (response.IsSuccessful) {
          if (response.IsVerified) {
            alert('Все подписи успешно проверены.');
          } else {
            alert('Одна или несколько подписей не прошли проверку.');
          }
        } else {
          alert(response.ErrorMessage);
        }
      })
      .fail(function(error) {
          alert('Ошибка при вызове метода verifySign.');
      });
```

Функция getCertificateList

Возвращает массив действительных сертификатов, которые можно использовать для формирования электронной подписи.

Каждый сертификат представляет собой объект JSON, состав полей которого описан ниже.

Пример

```
  client.getCertificateList()
      .done(function(response) {
          var certs = response.Certificates.map(function(x) {
            return JSON.parse(x.CertificateJson);
          });
          $.each(certs, function(index, item) {
              var date = new Date(parseInt(item.NotAfter.replace('/Date(', '')));
              var options = {
               year: 'numeric',
               month: 'numeric',
               day: 'numeric',
              };
              var dateString = date.toLocaleString("ru", options);

              $('#certificates').append($('<option></option>')
.val(item.Base64RawData)
.html(item.Subject + " до " + dateString));
          });

      });
```

Сертификаты в формате JSON

В результате выполнения некоторых функций программа ViPNet PKI Client Web Unit может возвращать сертификат в формате JSON. В этом случае в состав сертификата входят следующие поля:

- `certificate.Extensions` — расширения сертификата.
- `certificate.FriendlyName` — понятное имя сертификата.
- `certificate.Issuer` — издатель сертификата.

- `certificate.NotAfter` — конец срока действия сертификата.
- `certificate.NotBefore` — начало срока действия сертификата.
- `certificate.Base64RawData` — сертификат в формате для экспорта.
- `certificate.SerialNumber` — серийный номер сертификата.
- `certificate.SignatureAlgorithm` — алгоритм электронной подписи.
- `certificate.Subject` — субъект сертификата.
- `certificate.Thumbprint` — отпечаток сертификата.
- `certificate.Version` — версия сертификата.
- `VerifyMask` (см. Значения константы VerifyMask при проверке подписи на стр. 72) — константа с кодом состояния сертификата.

Коды возврата ошибок

В таблице ниже приведено соответствие кодов ошибок (ErrorCode) и сообщений об ошибках (ErrorMessage), которые могут возникнуть в ходе выполнения запросов. Язык текста ошибки зависит от языка локализации ОС.

Таблица 4. Коды ошибок запросов

Код	Текст ошибки (английский)	Текст ошибки (русский)
0	null	Запрос выполнен успешно
1	Failed to parse input request	Ошибки при разборе входного запроса
2	Service is busy.	Сервис занят
3	ViPNet PKI Client is temporary locked	ViPNet PKI Client временно заблокирован
4	User canceled the requested operation	Пользователь отклонил выполнение запрошенной операции
5	Base64 decode failed	Ошибка декодирования данных в формате Base64
6	Out of memory	Недостаточно памяти
7	Unknown type of request	Неизвестный тип запроса
8	Request is not available	Запрос не доступен
9	Failed to open default certificate store	Не удалось открыть хранилище сертификатов по умолчанию
10	Failed to find a private key container is required decrypting the message	Не удалось найти требуемый контейнер закрытого ключа необходимый для расшифровки сообщения
11	Private key for decrypt not found	Закрытый ключ для расшифровки не найден
12	Decrypt error	Ошибка при расшифровании сообщения

Код	Текст ошибки (английский)	Текст ошибки (русский)
13	User refused to enter the password for the key container	Пользователь отклонил ввод пароля для доступа к контейнеру закрытого ключа
14	Verify signature error	Ошибка при проверке подписи
15	Encrypt error	Ошибка при шифровании сообщения
16	Failed to open certificate store	Не удалось открыть хранилище сертификатов
17	Sign error	Ошибка при формировании подписи
18	Adding timestamp error	Ошибка при добавлении штампа времени
19	Signing certificate is not set or incorrect.	Сертификат для подписи не указан или поврежден
20	Signing certificate is not set.	Сертификат для подписи не указан
21	Error generating PKCS10 request	Ошибка при создании запроса PKCS10 на сертификат
22	Private key container not found	Контейнер ключа электронной подписи не найден
23		Введен некорректный пароль
24		Файл не является корректным xml документом
25		Непредвиденная ошибка

Значения константы VerifyMask при проверке подписи

Константа `VerifyMask` — может принимать одно или несколько значений, приведенных в таблице ниже.

Таблица 5. Коды состояний при проверке сертификата

Код	Описание
0	Сертификат успешно прошел проверку
1	Срок действия сертификата истек или не наступил
2	Сертификат отозван
4	Ошибка построения цепочки сертификатов
8	Срок действия закрытого ключа истек или не наступил

3

Проблемы и неисправности

При запуске Web Unit появляется сообщение о том, что порт занят

Если при запуске программы Web Unit появляется сообщение о недоступности порта 61111 или 61112, выполните следующие действия:

1. Убедитесь, что программа Web Unit не была запущена ранее другим пользователем вашего компьютера.
2. Если программа запущена другим пользователем, попросите этого пользователя выполнить вход в систему под своей учетной записью и завершить работу программы Web Unit. Затем выполните вход в систему под своей учетной записью и перезапустите Web Unit.
3. Если программа не запущена, с помощью стандартной консольной утилиты `netstat` проверьте, не используют ли порт 61111 или 61112 другие программы, запущенные на вашем компьютере.

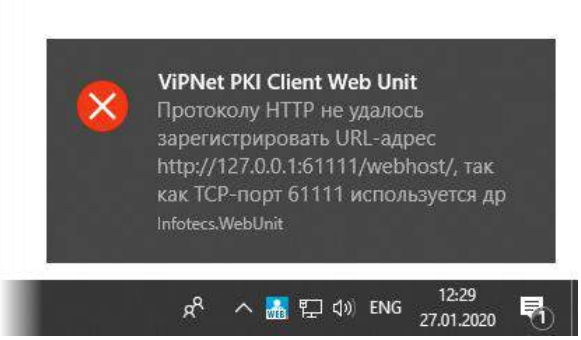

Рисунок 19. Ошибка, связанная с недоступностью порта

При попытке заверить документ электронной подписью появляется пустой список сертификатов

Если при попытке заверить документ электронной подписью список сертификатов пуст, убедитесь, что выполнены следующие требования:

- Сертификат должен быть действителен:
  - Срок действия сертификата не истек.
  - Сертификат не находится в списке аннулированных сертификатов доверенного удостоверяющего центра.
  - Вся цепочка сертификации полна, и все входящие в нее сертификаты удостоверяющих центров действительны.
  - В случае если запрос на сертификат был создан не с помощью ViPNet PKI Client или ViPNet CSP, сертификат должен быть установлен в контейнер (см. глоссарий, стр. 81) с соответствующим закрытым ключом (см. документ «ViPNet CSP 4.2. Руководство пользователя», раздел «Установка сертификата в контейнер ключей»).
- Сертификат должен иметь назначение **Цифровая подпись** в поле **Использование ключа** и должен быть установлен в системное хранилище **Личное**.

Внимание! В случае если сертификат не соответствует указанным требованиям, вы не сможете выбрать его для заверения электронной подписью.

При попытке подключения по протоколу HTTPS веб-браузер Mozilla Firefox выдает ошибку

Если вы используете веб-браузер Mozilla Firefox и при попытке подключения к IP-адресу `https://127.0.0.1:61112/` или `https://127.0.0.1:62223/` появляется страница **Это соединение является недоверенным**, добавьте соответствующий IP-адрес в исключения. Для этого выполните следующие действия:

1 На странице **Это соединение является недоверенным** в разделе **Я понимаю риск** нажмите кнопку **Добавить исключение**.

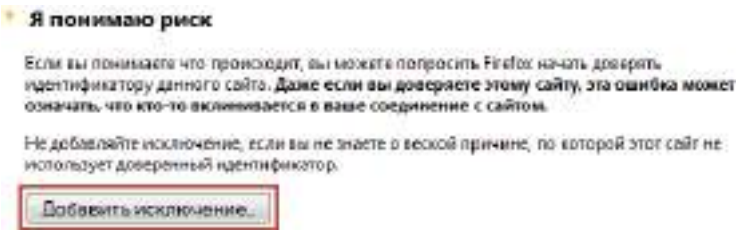

Рисунок 20. Вызов окна добавления исключения безопасности в веб-браузере Mozilla Firefox

2 В окне **Добавить исключение безопасности** нажмите кнопку **Подтвердить исключение безопасности**.

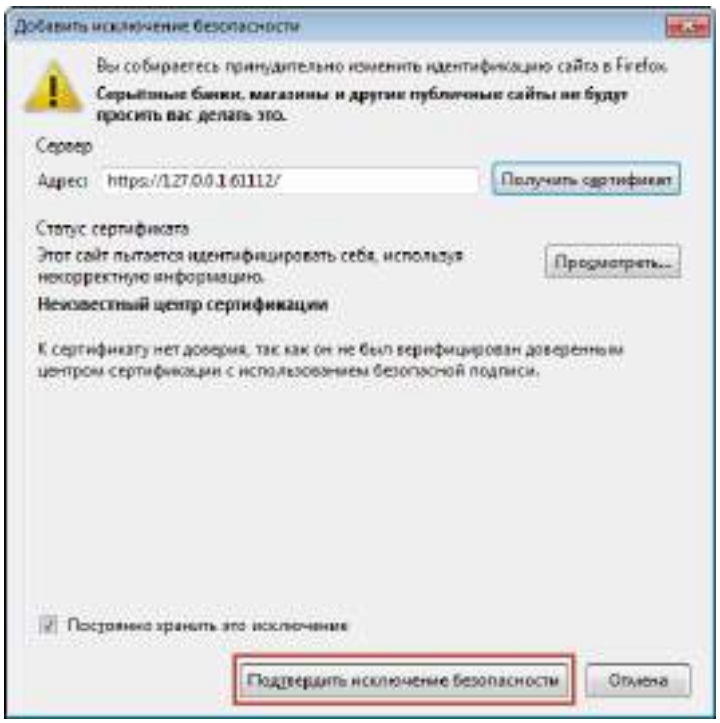

Рисунок 21. Добавление исключения безопасности в веб-браузере Mozilla Firefox

После добавления указанного исключения безопасности при обращении к программе Web Unit в веб-браузере Mozilla Firefox окно с ошибкой более не появится.

При попытке заверить данные электронной подписью веб-браузер Internet Explorer выдает ошибку

Если для доступа к веб-приложению вы используете веб-браузер Internet Explorer и при попытке заверить данные электронной подписью появляется окно с ошибкой **Объект не поддерживает свойство или метод**, выполните следующие действия:

1. В браузере Internet Explorer в меню **Сервис** выберите пункт **Параметры просмотра режима совместимости**.
2. В открывшемся окне выполните следующие действия:
   - Убедитесь, что URL-адрес веб-приложения, с помощью которого вы пытаетесь подписать данные, отсутствует в списке **Веб-сайты, для которых вы выбрали просмотр в режиме совместимости**. В противном случае выберите его в списке и нажмите кнопку **Удалить**.
   - Убедитесь, что снят флажок **Отображать сайты интрасети в режиме совместимости**.

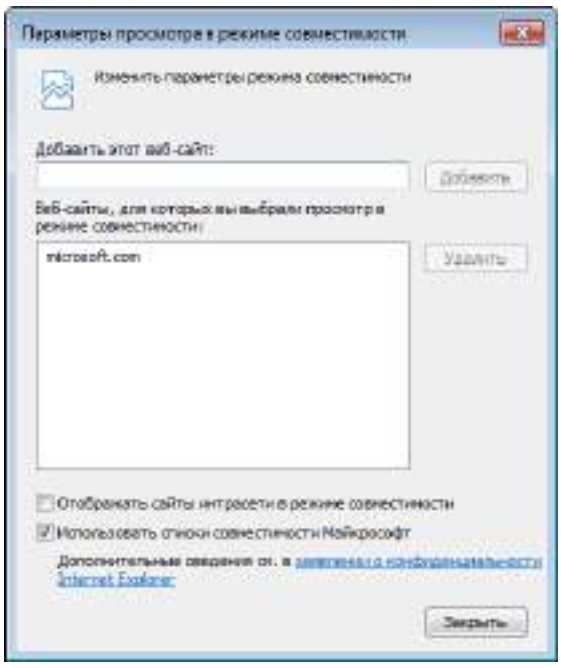

Рисунок 22. Настройка параметров режима совместимости

После описанной настройки параметров режима совместимости при обращении к программе Web Unit в веб-браузере Internet Explorer окно с ошибкой более не появится.

При работе по протоколу HTTPS Web Unit выдает сообщение об ошибке

Для выполнения криптографических операций при установленном соединении по протоколу HTTPS программой Web Unit используется самоподписанный сертификат, который автоматически создается и устанавливается в системное хранилище в процессе установки программы. Если срок действия этого сертификата истек, то после подключения по протоколу HTTPS при попытке выполнения криптографических операций появляется сообщение **Не удалось отправить файл на файл-сервер**.

Чтобы устранить эту ошибку, переустановите ПК ViPNet PKI Client.

A

Глоссарий

PKI (Public Key Infrastructure)

Инфраструктура открытых ключей — комплекс аппаратных и программных средств, политик и процедур, обеспечивающих распространение доверительного отношения к открытым ключам (в том числе ключам проверки электронной подписи) в распределенных системах через создание сертификатов ключей проверки электронной подписи и поддержание их жизненного цикла.

Асимметричное шифрование

Система шифрования, при которой алгоритмы используют два математически связанных ключа. Открытый ключ используется для зашифрования и передается по незащищенному каналу. Закрытый ключ служит для расшифрования.

Ключ проверки электронной подписи

В соответствии с федеральным законом N 63-ФЗ «Об электронной подписи» от 6 апреля 2011 г. ключом проверки электронной подписи называется открытый ключ, который является не секретной частью пары асимметричных ключей и представляет собой уникальную последовательность символов, однозначно связанную с закрытым ключом и предназначенную для проверки подлинности электронной подписи.

Ключ электронной подписи

В соответствии с федеральным законом N 63-ФЗ «Об электронной подписи» от 6 апреля 2011 г. ключом электронной подписи называется закрытый ключ, который является секретной частью пары асимметричных ключей и представляет собой уникальную последовательность символов, предназначенную для создания электронной подписи.

Контейнер ключей

Файл или устройство, в котором хранятся ключ электронной подписи и соответствующий ему сертификат ключа проверки электронной подписи.

Открепленная подпись

Тип электронной подписи, при использовании которого электронная подпись и служебная информация помещаются в отдельный контейнер `<имя_файла>.detached.sig`. Для проверки электронной подписи требуется не только данный контейнер, но и исходный файл, который в контейнер не входит.

Прикрепленная подпись

Тип электронной подписи, при использовании которого исходный файл, электронная подпись и служебная информация помещаются совместно в один контейнер. Далее для проверки электронной подписи требуется только данный контейнер, который содержит и электронную подпись, и исходный файл.

Сертификат ключа проверки электронной подписи

Электронный документ или документ на бумажном носителе, выданный удостоверяющим центром либо доверенным лицом удостоверяющего центра и подтверждающий принадлежность ключа проверки электронной подписи владельцу сертификата ключа проверки электронной подписи.

Удостоверяющий центр

Организация, осуществляющая выпуск сертификатов ключей проверки электронной подписи, а также сертификатов другого назначения.