Создание и проверка подписи происходят на стороне пользователя. При создании подписи с помощью **КриптоПро ЭЦП Browser plug-in**, электронная подпись может быть либо добавлена к подписываемым данным (присоединенная ЭП), либо создана отдельно (отделенная ЭП). ----- <http://www.cryptopro.ru/products/cades/plugin>

Отделенная подпись создается посредством указания параметра bDetached = TRUE метода [Sign](http://msdn.microsoft.com/en-us/library/aa387726.aspx" \o "http://msdn.microsoft.com/en-us/library/aa387726.aspx) или [SignCades](http://www.cryptopro.ru/docs/cpdn/default.asp?url=content/cades/interface_c_ad_e_s_c_o_m_1_1_i_c_p_signed_data2_2baa8aa41b13d58731e57667ead5ab1c_12baa8aa41b13d58731e57667ead5ab1c.html" \o "http://www.cryptopro.ru/docs/cpdn/default.asp?url=content/cades/interface_c_ad_e_s_c_o_m_1_1_i_c_p_signed_data2_2baa8aa41b13d58731e57667ead5ab1c_12baa8aa41b13d58731e57667ead5ab1c.html) объекта CadesSignedData.  
  
При этом возвращаемое этими методами значение и будет представлять собой отделенную подпись.  
  
Проверять такую подпись следует предварительно передав данные в свойство Content объекта CadesSignedData.  
  
В плагине подпись осуществляется на том алгоритме, который указан в сертификате, переданном при ее создании. См. также [примечание](http://www.cryptopro.ru/docs/cpdn/default.asp?url=content/cades/class_c_ad_e_s_c_o_m_1_1_cades_signed_data.html).

[*Отредактировано пользователем*](http://www.cryptopro.ru/forum2/default.aspx?g=posts&t=3730) 5 лет назад | *Причина: Не указана*

В отличие от объекта CAPICOM.SignedData ,   
методы Sign, CoSign и Verify создают и проверяют усовершенствованную подпись типа CAdES-X Long Type 1.

подписи CAdES BES, CADES-T и CADES-X LONG TYPE 1

// Создание объектов КриптоПро ЭЦП Browser plug-in

var oStore = cadesplugin.CreateObject("CAdESCOM.Store");

var oSigner = cadesplugin.CreateObject("CAdESCOM.CPSigner");

var oPrivateKey = cadesplugin.CreateObject("X509Enrollment.CX509PrivateKey");

## Почему моя отделённая подпись из плагина не проверяется в КриптоАРМ/cryptcp

* [КриптоПро ЭЦП](https://www.cryptopro.ru/category/faq/produkty/kriptopro-etsp)

Опубликовано een в 04 Октябрь 2011 - 13:49

В КриптоПро ЭЦП Browser plug-in при создании электронной подписи происходит перекодирование исходных данных в UCS-2 little endian и именно в таком виде данные будут подписаны. Это сделано для совместимости с теми ЭП, которые были созданы в Internet Explorer с использованием CAPICOM.  
В КриптоАРМ и в cryptcp перекодирование данных при создании подписи не производится.  
Для того, чтобы данные для подписи в КриптоПро ЭЦП Browser plug-in не перекодировались в UCS-2 little endian, их нужно передавать следующим образом:

1. задать значение свойства [ContentEncoding](http://cpdn.cryptopro.ru/default.asp?url=content/cades/interface_c_ad_e_s_c_o_m_1_1_i_c_p_signed_data3_c8fe269303436af3d6dd0ea4ba4e6412_1c8fe269303436af3d6dd0ea4ba4e6412.html) = BASE64\_TO\_BINARY;
2. перекодировать исходные бинарные данные в BASE64;
3. передать закодированные в BASE64 данные в свойство Content;
4. при создании подписи будут подписаны исходные бинарные данные.

В разделе **Документация** на нашем сайте расположен [пример создания и проверки отделенной подписи, которая будет успешно проверена в КриптоАРМ/Cryptcp](http://cpdn.cryptopro.ru/default.asp?url=content/cades/plugin-samples-sign-detached.html).  
Более подробная информация об особенностях работы с бинарными данными в КриптоПро ЭЦП Browser plug-in [здесь](http://cpdn.cryptopro.ru/default.asp?url=content/cades/content-encoding.html).

var CADESCOM\_CADES\_BES = 1;

var CAPICOM\_CURRENT\_USER\_STORE = 2;

var CAPICOM\_MY\_STORE = "My";

var CAPICOM\_STORE\_OPEN\_MAXIMUM\_ALLOWED = 2;

var CAPICOM\_CERTIFICATE\_FIND\_SUBJECT\_NAME = 1;

var CADESCOM\_BASE64\_TO\_BINARY = 1;

function SignCreate(certSubjectName, dataToSign) {

var oStore = cadesplugin.CreateObject("CAPICOM.Store");

oStore.Open(CAPICOM\_CURRENT\_USER\_STORE, CAPICOM\_MY\_STORE,

CAPICOM\_STORE\_OPEN\_MAXIMUM\_ALLOWED);

var oCertificates = oStore.Certificates.Find(

CAPICOM\_CERTIFICATE\_FIND\_SUBJECT\_NAME, certSubjectName);

if (oCertificates.Count == 0) {

alert("Certificate not found: " + certSubjectName);

return;

}

var oCertificate = oCertificates.Item(1);

var oSigner = cadesplugin.CreateObject("CAdESCOM.CPSigner");

oSigner.Certificate = oCertificate;

var oSignedData = cadesplugin.CreateObject("CAdESCOM.CadesSignedData");

// Значение свойства ContentEncoding должно быть задано

// до заполнения свойства Content

oSignedData.ContentEncoding = CADESCOM\_BASE64\_TO\_BINARY;

oSignedData.Content = dataToSign;

var sSignedMessage = "";

try {

sSignedMessage = oSignedData.SignCades(oSigner, CADESCOM\_CADES\_BES, true);

} catch (err) {

alert("Failed to create signature. Error: " + cadesplugin.getLastError(err));

return;

}

oStore.Close();

return sSignedMessage;

}

function Verify(sSignedMessage, dataToVerify) {

var oSignedData = cadesplugin.CreateObject("CAdESCOM.CadesSignedData");

try {

// Значение свойства ContentEncoding должно быть задано

// до заполнения свойства Content

oSignedData.ContentEncoding = CADESCOM\_BASE64\_TO\_BINARY;

oSignedData.Content = dataToVerify;

oSignedData.VerifyCades(sSignedMessage, CADESCOM\_CADES\_BES, true);

} catch (err) {

alert("Failed to verify signature. Error: " + cadesplugin.getLastError(err));

return false;

}

return true;

}

function run() {

var oCertName = document.getElementById("CertName");

var sCertName = oCertName.value; // Здесь следует заполнить SubjectName сертификата

if ("" == sCertName) {

alert("Введите имя сертификата (CN).");

return;

}

// Предварительно закодированные в BASE64 бинарные данные

// В данном случае закодирована строка "Some Data."

var dataInBase64 = "U29tZSBEYXRhLg==";

// Подписаны будут исходные бинарные данные (в данном случае - "Some Data.")

// Такая подпись должна проверяться в КриптоАРМ и cryptcp.exe

var signedMessage = SignCreate(sCertName, dataInBase64);

document.getElementById("signature").innerHTML = signedMessage;

var verifyResult = Verify(signedMessage, dataInBase64);

if (verifyResult) {

alert("Signature verified");

}

}

<http://pushorigin.ru/net/start>

http://cpdn.cryptopro.ru/content/cades/plugin-methods.html

<http://cpdn.cryptopro.ru/default.asp?url=content/cades/plugin.html>

демо

<https://www.cryptopro.ru/sites/default/files/products/cades/demopage/cades_bes_file.html>

interface COM

<http://cpdn.cryptopro.ru/content/cades/cadescom.html>

Добрый день!  
  
Столкнулся со следующей проблемой:  
Есть XML файл (запрос на получение реестра запрещённых сайтов, будь он неладен Brick wall )  
Создаю отсоединенную подпись с помощью cryptcp (windows).   
cryptcp.exe -signf -der request.xml  
Проверяю её на [госуслугах](http://www.gosuslugi.ru/pgu/eds" \o "http://www.gosuslugi.ru/pgu/eds) - "Не найден сертификат подписи", "Подлинность документа НЕ ПОДТВЕРЖДЕНА".  
Естественно, и роскомнадзор запрос в таком виде не принимает.  
  
Но с сертификатом всё нормально!  
Создаю подпись в файле:  
cryptcp.exe -sign -der request.xml  
И он проходит проверку! "Проверка выполнена", "Подлинность сертификата ПОДТВЕРЖДЕНА"  
  
Подскажите, пожалуйста - где собака порылась? Что я делаю неправильно?...  
  
PS:  
КриптоПРО CSP 3.6, куплен неделю назад  
cryptcp скачан вчера  
т.е. софт свежий.

2.4. Работа с пакетами файлов  
...  
[-dir <папка>] -signf <КПС1> <маска файлов> [-cert] [-crl] [-der]  
[-sd[<URL>]] [-ss[<URL>]] [-nostampcert] [-pin <пароль>|-askpin]  
Создать подписи файлов и записать их в файлы “имя\_исходного\_файла.sgn”.   
...  
**-cert** – **добавлять** в подписи **сертификат отправителя**;

Добрый день!  
Правильно ли я понимаю механизм проверки:  
1. из подписанного файла берется сертификат  
2. сертификат проверяется по цепочке  
3. на основе сертификата вычисляется хеш  
4. хеш сравнивается собственно с подписью в подписанном файле  
  
То есть если не приложить сертификат в подписываемый документ, и на машине также нет этого сертификата, то документ проверку не пройдет?  
Иными словами сертификаты ищутся в самом подписанном документе, а также в хранилище?  
  
Спасибо

Добрый день.  
  
Из сертификата - извлекается открытый ключ, который нужен для проверки математической корректности электронной подписи.  
Сертификат может быть включен в файл с эл.подписью или быть в другом источнике (файл\хранилище сертификатов) - зависит от ПО.  
Обычно сертификат включают в файл с эл.подписью.  
  
Далее, после успешной проверки эл.подписи - проверяется сам сертификат (срок действия\путь сертификации\проверка статуса (CRL\OCSP)\полномочия владельца и т.д.).

КриптоАРМ умеет сам определять это.  
  
Прикрепленная подпись - подпись + данные в одном файле, после проверки подписи - можно извлечь подписанный документ на диск.  
  
Отделенная подпись - подпись и данные в разных файлах - для проверки требуется исходный документ + подпись (небольшие файлы:.p7s, .sig, .sgn)

Не проверяются чужие подписи

https://www.cryptopro.ru/forum2/default.aspx?g=posts&t=9814

Разные расширения могут быть, зависит от требований конкретной информационной системы, примеры:  
.p7s  
.sig  
.sign  
.sgn

в документации налажали с примером  
**нельзя ли привести к единому виду создания объектов?**  
  
**что это за сюрприз**(?)  
var oSigner = CreateObject("CAdESCOM.CPSigner");  
НО при этом  
var oStore = CreateObject("**CAPICOM**.Store");  
  
**как-то топорная работа**

Читая более внимательно:

**Цитата:**

CAdESCOM.Store  
[http://cpdn.cryptopro.ru...t/cades/class\_store.html](http://cpdn.cryptopro.ru/content/cades/class_store.html)  
Замечание:  
В CAdESCOM данный объект доступен только на платформе x64.  
На платформе win32 следует использовать объект Microsoft CAPICOM.Store .

var oSigner = CreateObject("CAdESCOM.CPSigner");

НО при этом

var oStore = CreateObject("CAPICOM.Store");

Это сделано специально, чтобы один и тот же код одинаково работал в разных браузерах на разных платформах.  
  
В 32-битном IE будет создаваться объект CAPICOM.Store (из "родного" 32-битного CAPICOM).  
Так как 64-битного CAPICOM нет, то данный объект для платформы х64 реализован в CAdESCOM.dll.  
Идентификатор оставлен прежним, чтобы не приходилось программно проверять браузер и платформу.

**Цитата:**

поддерживается ли возможность использовать CADESCOM с алгоритмами НЕ ГОСТ ? (например есть острая необходимость хэш делать sha1, но подписывать с сертификатом gost

CAdESCOM не подразумевает использование какого-то конкретного алгоритма.  
  
А вот стандарт создания электронной подписи ГОСТ Р 34.10-2001 требует, чтобы хэш-значение сообщения было вычислено в соответствии с ГОСТ Р 34.11-94.  
  
Или вы имеете в виду какое-то другое хэш-значение?----

<https://ru.wikipedia.org/wiki/CAdES>

А как быть, с отделенной подписью, если расшифрованные данные не совпали с передаваемыми и было сгенерировано исключение? Откуда я могу получить список подписавших? Спасибо.

с помощью MS CryptoAPI можно получить список и информацию о сертификатах

|  |
| --- |
| . Как я понимаю, речь идет о подписи произвольного бинарного файла.  В этом случае для совместимости с КриптоАРМ рекомендуем перекодирование файла в BASE64 (только для создания подписи в плагине) и использование параметра CADESCOM\_BASE64\_TO\_BINARY в качестве значения свойства [CadesSignedData.ContentEncoding](http://www.cryptopro.ru/docs/cpdn/default.asp?url=content/cades/interface_c_ad_e_s_c_o_m_1_1_i_c_p_signed_data3_c8fe269303436af3d6dd0ea4ba4e6412_1c8fe269303436af3d6dd0ea4ba4e6412.html" \o "http://www.cryptopro.ru/docs/cpdn/default.asp?url=content/cades/interface_c_ad_e_s_c_o_m_1_1_i_c_p_signed_data3_c8fe269303436af3d6dd0ea4ba4e6412_1c8fe269303436af3d6dd0ea4ba4e6412.html). Данный параметр должен быть указан до того, как будет присвоено или получено значение свойства Content.   Для проверки в КриптоАРМ исходный бинарный файл перекодировать не надо.  То есть: 1. Сначала перекодируете исходный файл в BASE64. 2. Указываете signedData.ContentEncoding = CADESCOM\_BASE64\_TO\_BINARY. 3. Заполняете signedData.Content = закодированный в BASE64 файл. 4. Подписываете.  Получаете подпись, где подписанные данные = содержимое исходного файла. Эта подпись должна проверяться в КриптоАРМ без какого-либо дополнительного перекодирования.    2. При проверке подписи в плагине оригинал документа нужно передавать в свойство Content. Кодировка, в которой вы передаете оригинал документа, должна быть указана заранее.  Для бинарных файлов (см. п.1) алгоритм должен быть таким:  1. Указываете signedData.ContentEncoding = CADESCOM\_BASE64\_TO\_BINARY. 2. Заполняете signedData.Content = закодированный в BASE64 исходный файл. 3. Проверяете signedData.Verify … |
| [Вверх](javascript:%20void(0)) | |  |
|  |  |  |

<https://github.com/nicolaskruchten/pivottable>

CadesSignedData.Signers.Item(номер подписи).[SigningTimestampTime](http://www.cryptopro.ru/docs/cpdn/default.asp?url=content/cades/interface_c_ad_e_s_c_o_m_1_1_i_c_p_signer3_3d992ac762785a12157f14393912fd03_13d992ac762785a12157f14393912fd03.html)- время в штампе.

отсюда **samples/samples\_src.jar/userSamples/SignAndVerify.java**

**------------------**

#### Как нам пришлось помучиться

EUC-KR → KOI8-R

<http://www.cryptopro.ru/forum2/default.aspx?g=posts&m=77345#post77345>

Про лицензии серверная клентская <https://www.cryptopro.ru/forum2/default.aspx?g=posts&t=5362>

Перечитать интереснно

http://www.cryptopro.ru/forum2/?g=posts&t=3676

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | 13.06.2006 10:49:39 | **КриптоПро JCP в чем разница между лицензиями** |  | Ответов: 1 | |
| |  |  |  | | --- | --- | --- | | https://cryptopro.ru/cryptopro/forum/images/author.gif Евгений |  |  | |
| Прошу пояснить чем отличаються лицензии: - Лицензия на право пользования СКЗИ КриптоПро JCP на одном рабочем месте - Лицензия (серверная) на право пользования СКЗИ КриптоПро JCP на одном сервере с 1 процессором  т.е. какие ограничения накладывает пользовательская лицензия на рабочее место по сравнению с серверной?  PS: почему то не смог обнаружить это в документации :( |
|  |
| Ответы: |
| |  |  | | --- | --- | | 13.06.2006 13:26:36 | Юрий Маслов | |
| Лицензия на право пользования СКЗИ КриптоПро JCP на одном рабочем месте - исходя из своего названия дает право пользования КриптоПро JCP на рабочем месте. А серверная лицензия - на сервере. Поэтому неправильно говорить чем они отличаются. Для работы нужно покупать и ту и другую. Ведь клиента без сервера быть не может, равно как и сервера без клиентов держать не целесообразно :-) Поэтому покупают 1 или несколько серверных лицензий ну и клиентских лицензий столько, сколько нужно |

Здравствуйте,  
подскажите пожалуйста, я правильно понимаю, что используя в качестве типа подписи CAdES-BES или CAdESXLongType1 в любом случае получается **усиленная квалифицированная электронная подпись**, а выбор между CAdES-BES/CAdESXLongType1 состоит в уровне защиты. То есть при использовании подписи типа CAdESXLongType1 получаете больший уровень защиты нежели используя CAdES-BES в виду наличия доказательств подписи у CAdESXLongType1. Верно?

О эдо

[**http://ecm-journal.ru/e-sign#four**](http://ecm-journal.ru/e-sign#four)

[**Подписание множества объектов с единократным вводом пароля**](https://www.cryptopro.ru/forum2/default.aspx?g=posts&t=11165)

[**https://www.cryptopro.ru/forum2/default.aspx?g=topics&f=9&p=3**](https://www.cryptopro.ru/forum2/default.aspx?g=topics&f=9&p=3)

Здравствуйте!  
Есть необходимость проверять ЭЦП, которые могут быть созданны сертификатом любого из аккредитованных УЦ. Соответственно нужно установить сертификаты всех этих УЦ. Правильно ли я понимаю, что сертификаты аккредитованных УЦ надо устанавливать в промежуточное хранилище, а сертификат ГУЦ в корневое?

Правильно.

cert.Export(0)  
выводит длинный хеш вида:  
"MIIJ9TCCCaSgAwIBAgIKGt/ukQACAAFjITAIBgYqhQMCAgMwggE8MRgwFgYFKoUD  
ZAESDTEwOTc3NDYyOTkzNTMxGjAYBggqhQMDgQMBARIMMDA3NzA3NzA0NjkyMTww  
OgYDVQQJDDPQv9GA0L7RgdC/0LXQutGCIDYwLdC70LXRgtC40Y8g0J7QutGC0Y/Q

Это оно и есть. Только в кодировке base64. Все нормальные программы смогут его переварить в таком виде. Если очень хочется покороче, то надо декодировать из base64.

По разделению

[**https://www.cryptopro.ru/forum2/default.aspx?g=posts&t=6512**](https://www.cryptopro.ru/forum2/default.aspx?g=posts&t=6512)

sSignedMessage - это я так понимаю данные из файла подписи?! Правильно?

sSignedMessage - в данном случае данные с присоединенной ЭЦП  
  
Если ЭЦП отдельная - необходимо заполнить Content  
oSignedData.**Content** = verifyData - исходные данные, которые были подписаны   
  
oSignedData.**VerifyCades ( ... )**

А если нет исходных данных, есть только файл подписи, тогда как быть?

только файл с отсоединенной подписью?   
А цель какая? Узнать кто, когда, что-то подписал, не имея это "что-то" на руках?  
Тогда заполнить .content "любыми данными"... и *обработав исключение*, читать информацию о Signer(s)  
  
или имеется ввиду все таки "есть только файл... с присоединенной ЭЦП"? Тогда - пример с sSignedMessage выше приведен.