Цифровые технологии

КриптоАРМ Server

Документация продукта

Релиз v1.0

> Архивная документация. Актуальный релиз: v1.3

Исходная документация на CryptoARM.ru

Редакция от 26.11.2026

Правообладатель и разработчик
ООО «Цифровые технологии»

Содержание

КриптоАРМ Сервер 10
О продукте 12
Инструкции по установке 14
Примеры использования
Автоматизация проверки электронной подписи с использованием КриптоАРМ Сервер 21
Автоматическое подписание обезличенным сертификатом 26
КриптоАРМ. Возможности по интеграции и автоматизации 30
Интеграция КриптоАРМ во внешнюю информационную систему 34
Сервис проверки и визуализации электронной подписи 39
Работа с почтой в Node.js. Примеры и возможности КриптоАРМ Сервер 42
Сервис проверки и улучшения электронной подписи 46
FAQ
Часто задаваемые вопросы 54
Глоссарий 58
Руководство программиста (SDK)
Описание API модуля trusted-crypto 64
Пространство имен CMS
Класс SignedData
Описание класса SignedData 65
Метод load 66
Метод sign 67
Метод import 68
Метод export 69
Метод save 70
Метод verify 71
Метод content 72
Метод policies 73
Метод freeContent 74
Метод isDetached 75
Метод certificates 76

Метод signers 77
Метод signParams 78

Класс Signer

Описание класса Signer 79
Метод certificate 80
Метод index 81
Метод signingTime 82
Метод signatureAlgorithm 83
Метод signatureDigestAlgorithm 84
Метод issuerName 85
Метод issuerName 86
Метод timestamp 87
Метод verifyTimestamp 88
Метод isCades 89
Метод certificateValues 90
Метод revocationValues 91
Метод ocspResp 92
Метод verifySignatureValue 93

Класс SignerCollection

Описание класса SignerCollection 94
Метод items 95
Метод length 96

Класс CadesParams

Описание класса CadesParams 97
Метод cadesType 98
Метод connSettings 99
Метод tspHashAlg 100
Метод ocspSettings 101

Класс TimestampParams

Описание класса TimestampParams 102
Метод connSettings 103
Метод tspHashAlg 104
Метод ocspSettings 105

Перечисляемые типы
SignedDataContentType 106
StampType 107
CadesType 108
Интерфейсы
ISignedDataContent 109
Примеры 110
Пространство имен PKI
Класс OID
Описание класса OID 113
Метод value 114
Метод longName 115
Метод shortName 116
Примеры 117
Класс Extension
Описание класса Extension 118
Метод typeId 119
Метод critical 120
Примеры 121
Класс ExtensionCollection
Описание класса ExtensionCollection 122
Метод items 123
Метод length 124
Метод push 125
Метод pop 126
Метод removeAt 127
Примеры 128
Класс CRL
Описание класса CRL 129
Метод load 130
Метод import 131
Метод version 132

Метод issuerFriendlyName ... 133
Метод issuerName ... 134
Метод lastUpdate ... 135
Метод nextUpdate ... 136
Метод thumbprint ... 137
Метод signatureAlgorithm ... 138
Метод signatureDigestAlgorithm ... 139
Метод authorityKeyid ... 140
Метод crlNumber ... 141
Метод compare ... 142
Метод equals ... 143
Метод hash ... 144
Метод duplicate ... 145
Метод export ... 146
Метод save ... 147
Примеры ... 148

Класс CrlCollection

Описание класса CrlCollection ... 149
Метод items ... 150
Метод length ... 151
Метод push ... 152
Метод pop ... 153
Метод removeAt ... 154
Примеры ... 155

Класс Certificate

Описание класса Certificate ... 156
Метод load ... 158
Метод import ... 159
Метод version ... 160
Метод serialNumber ... 161
Метод keyUsage ... 162
Метод issuerFriendlyName ... 163
Метод issuerName ... 164

Метод subjectFriendlyName ... 165
Метод subjectName ... 166
Метод notBefore ... 167
Метод notAfter ... 168
Метод thumbprint ... 169
Метод signatureAlgorithm ... 170
Метод signatureDigestAlgorithm ... 171
Метод publicKeyAlgorithm ... 172
Метод organizationName ... 173
Метод OCSPUrls ... 174
Метод CAIssuersUrls ... 175
Метод isSelfSigned ... 176
Метод isCA ... 177
Метод sign ... 178
Метод compare ... 179
Метод equals ... 180
Метод hash ... 181
Метод duplicate ... 182
Метод export ... 183
Метод save ... 184
Примеры ... 185

Класс CertificateCollection

Описание класса CertificateCollection ... 186
Метод items ... 187
Метод length ... 188
Метод push ... 189
Метод pop ... 190
Метод removeAt ... 191
Примеры ... 192

Класс CertificationRequest

Описание класса CertificationRequest ... 193
Метод subject ... 194
Метод version ... 195

Метод extensions ... 196
Метод containerName ... 197
Метод pubKeyAlgorithm ... 198
Метод exportableFlag ... 199
Метод newKeysetFlag ... 200
Метод save ... 201
Примеры ... 202

Класс Cipher

Описание класса Cipher ... 203
Метод ProvAlgorithm ... 204
Метод recipientsCerts ... 205
Метод encrypt ... 206
Метод decrypt ... 207
Примеры ... 208

Класс OCSP

Описание класса OCSP ... 209

Класс TSPRequest

Описание класса TSPRequest ... 210

Класс TSP

Описание класса TSP ... 211

Класс PKCS12

Описание класса PKCS12 ... 213
Метод load ... 214
Метод save ... 215
Примеры ... 216

Пространство имен utils

Класс Csp

Описание класса CSP ... 217
Метод enumContainers ... 219
Метод getCertificateFromContainer ... 220
Метод installCertificateFromContainer ... 221
Метод installCertificateToContainer ... 222

Метод deleteContainer ... 223
Метод getContainerNameByCertificate ... 224
Метод hasPrivateKey ... 225
Метод buildChain ... 226
Метод verifyCertificateChain ... 227
Метод isHaveExportablePrivateKey ... 228
Метод certToPkcs12 ... 229
Метод importPkcs12 ... 230
Примеры ... 231
Класс ConnectionSettings
Описание класса ConnectionSettings ... 232
Метод AuthType ... 234
Метод Address ... 235
Метод UserName ... 236
Метод Password ... 237
Метод ClientCertificate ... 238
Метод ProxyAuthType ... 239
Метод ProxyAddress ... 240
Метод ProxyUserName ... 241
Метод ProxyPassword ... 242
Класс ModuleInfo
Описание класса ModuleInfo ... 243
Класс Tools
Описание класса Tools ... 244
Пространство имен pkistore
Класс Filter
Описание класса Filter ... 245
Класс PkiStore
Описание класса PkiStore ... 246
Метод addProvider ... 247
Метод find ... 248
Метод getItem ... 249

Метод certs ... 250
Метод addCert ... 251
Метод addCert ... 252
Метод deleteCert ... 253
Метод deleteCrl ... 254
Класс ProviderCryptopro
Описание класса ProviderCryptopro ... 255
Интерфейсы
IPkiKey ... 256
IPkiCrl ... 257
IPkiRequest ... 258
IPkiCertificate ... 259
IPkiItem ... 260
Пространство имен common
Класс Logger
Описание класса Logger ... 261
Метод start ... 262
Дополнительные статьи
Руководство программиста (SDK) ... 263

Открыть статью на сайте

КриптоАРМ Сервер

КриптоАРМ Server — программный продукт, предназначенный для интеграции и автоматизации криптографических операций в информационных системах организаций. Он включает в себя набор криптографических библиотек и средства разработки (SDK), позволяющие встраивать функции шифрования данных, проверки электронных подписей и их создания в автоматизированных процессах обработки информации.

Функции продукта:

Проверка электронной подписи

- Автоматизируйте проверку электронной подписи для поступающих документов
- Создавайте и храните протоколы проверки электронной подписи
- Визуализируйте электронную подпись для PDF документов

Автоматизация электронной подписи

- Создайте свой собственный сервис электронной подписи
- Поддержка стандартов CMS, CAdES, PAdES
- Используйте КриптоАРМ на рабочих станциях и мобильных устройствах как клиент для электронной подписи

Защита исходящих документов

- Попробуйте сервис для отправки конфиденциальных документов
- Автоматическое шифрование файлов перед отправкой по e-mail
- Автоматическая обработка входящих писем: расшифрование, проверка подписи и категоризация

Продукт включает функциональные модули:

Модуль	Описание
trusted-crypto	Формирование электронной подписи, проверка подписи, шифрование и расшифрование, сведения о сертификатах, генерация ключевой пары и создание запросов на получение сертификата, подключение защищенных носителей.
trusted-pdf	Подпись и проверка электронной подписи встроенной в pdf, разметка областей для оттисков подписи, добавление графических штампов с информацией о подписи, пакетная подпись,

Модуль	Описание
	сертификация подписанных документов с защитой от изменений, конвертация pdf-файлов в формат pdf-a/2b.
trusted-mailcore	Сбор, разбор и отправка электронных писем, поддержка IMAP, SMTP, POP3, S/MIME, контакты и адресные книги vCard, CardDAV.
trusted-curl	Обеспечение защищенного соединения к веб-серверам с поддержкой ГОСТ TLS.
trusted-xml-crypto	Проверка электронной подписи формата XMLDSig (XML Digital Signature).

Открыть статью на сайте

О продукте

КриптоАРМ Server — программный продукт, предназначенный для интеграции и автоматизации криптографических операций в информационных системах организаций. Он включает в себя набор криптографических библиотек и средства разработки (SDK), позволяющие встраивать функции шифрования данных, проверки электронных подписей и их создания в автоматизированных процессах обработки информации.

Функции продукта:

Проверка электронной подписи

- Автоматизируйте проверку электронной подписи для поступающих документов
- Создавайте и храните протоколы проверки электронной подписи
- Визуализируйте электронную подпись для PDF документов

Автоматизация электронной подписи

- Создайте свой собственный сервис электронной подписи
- Поддержка стандартов CMS, CAdES, PAdES
- Используйте КриптоАРМ на рабочих станциях и мобильных устройствах как клиент для ЭП

Защита исходящих документов

- Попробуйте сервис для отправки конфиденциальных документов
- Автоматическое шифрование файлов перед отправкой по e-mail
- Автоматическая обработка входящих писем: расшифрование, проверка ЭП и категоризация

Продукт включает функциональные модули:

Модуль	Описание
trusted-crypto	Формирование электронной подписи, проверка подписи, шифрование и расшифрование, сведения о сертификатах, генерация ключевой пары и создание запросов на получение сертификата, подключение защищенных носителей.
trusted-pdf	Подпись и проверка электронной подписи встроенной в pdf, разметка областей для оттисков подписи, добавление графических штампов с информацией о подписи, пакетная подпись, сертификация подписанных документов с защитой от изменений, конвертация pdf-файлов в формат pdf-a/2b.

Модуль	Описание
trusted-mailcore	Сбор, разбор и отправка электронных писем, поддержка IMAP, SMTP, POP3, S/MIME, контакты и адресные книги vCard, CardDAV.
trusted-curl	Обеспечение защищенного соединения к веб-серверам с поддержкой ГОСТ TLS.
trusted-xml-crypto	Проверка электронной подписи формата XMLDSig (XML Digital Signature).

Открыть статью на сайте

Инструкции по установке

Условия применения

Требования для установки и настройки сервиса

Характеристика	Минимальное значение	Дополнение
Информационная емкость ОЗУ	4 Гб	
Информационная емкость HDD	50 Гб	
Процессор	x86_64 (или amd64)	
Поддерживаемые ОС	Ubuntu 18.04 LTS (Bionic Beaver), Ubuntu 20.04 LTS (Focal Fossa), Debian 11 (Bullseye), Astra Linux CE 2.12.45 (Orel)	Другие Linux дистрибутивы, поддерживающие запуск Docker
Дополнительно	Средство автоматизации развертывания и контейнеризации Docker	

Используемый стек технологий

Node.js, NestJS, КриптоПро CSP 5.0, КриптоАРМ ГОСТ SDK, Docker

Установка

Установка docker и docker-compose

Для установки проекта необходимо установить docker и docker-compose.
Подробное руководство по установке docker и docker-compose представлено на странице официальной документации проекта: https://docs.docker.com/engine/install/

Минимально необходимые версии компонентов Docker Engine:

- Docker 19.03 +

Установка и первый запуск сервиса

1. Скачайте архив server-main.zip с дистрибутивом программного продукта.

```
wget https://git.digtlab.ru/trusted/cryptoarm/server/-/archive/main/server-main.zip
```

2. Распакуйте скаченный архив

```
unzip server-main.zip
```

3. Войдите в каталог server-main:

```
cd server-main/
```

4. Скачайте КриптоПро CSP 5.0 для Linux (x64, deb) linux-amd64_deb.tgz в каталог cryptopro. Для скачивания требуется пройти авторизацию на сайте КриптоПро.

```
server-main/cryptopro/linux-amd64_deb.tgz
```

5. Отредактируйте настройки конфигурации:

- Откройте на редактирование файл **.env**;
- Задайте серверную лицензию КриптоПро CSP в переменной **CRYPTOPRO_LICENSE**
CRYPTOPRO_LICENSE=XXXXXXXXXXXXXXXXXXXXXXXXX
- Задайте серверную лицензию КриптоАРМ ГОСТ версии 3 в переменной **TRUSTED_LICENSE**
TRUSTED_LICENSE=XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

1. Запустите сборку проекта:

```
docker compose build --pull
```

2. Запустите проект:

```
docker compose up -d
```

Просмотреть логи:

```
docker compose logs -f
```

Остановить проект:

```
docker compose stop
```

Вход в сервис

Для входа в web интерфейс сервиса необходимо перейти по адресу **http://SERVER:PORT**. Пример: **http://172.17.1.115:3037**

Swagger документация по REST API доступна по адресу **http://SERVER:PORT/docs**. Пример: **http://172.17.1.115:3037/docs**

Проверка работы сервиса

Проверить работу сервиса можно следующими способами:

1. С помощью Swagger документации **http://SERVER:PORT/docs**
2. С помощью утилиты **curl**:

```
   curl -X 'POST' \
  'http://172.17.1.115:3037/cms/verify' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "cms": "MIAGCSqGSIb3...==,MSMwIQYJKoZI...==",
  "data": "dGVzdA=="
}'
```

В запросе передаются следующие параметры:

- **cms**: откреплennая (либо прикрепленная) подпись, закодированная в Base64
- **data**: подписанные данные в формате Base64 (для откреплennой подписи)

Пример ответа:

```
HTTP/1.1 201 Created
Vary: Origin
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
Content-Length: 136101
ETag: W/"213a5-+1/6qQ+hWCfJ0yGRz0/fQGpbTbk"
Date: Thu, 26 Sep 2024 12:40:04 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
  "status": 200,
  "message":"OK",
  "isValid": true,
  "isValidSign": true,
  "signs": [
    {
      "cadesTypeName": "CAdES-BES",
      "certificate": {
        "version": 3,
        "status": true,
        "subjectFriendlyName": "esia test3",
        "issuerFriendlyName": "CRYPTO-PRO Test Center 2",
        "subjectName": "CN=esia test3",
        "issuerName": "E=support@cryptopro.ru\r\nC=RU\r\nL=Moscow\r\nO=CRYPTO-PRO
LLC\r\nCN=CRYPTO-PRO Test Center 2",
        "notAfter": "2024-10-26T16:25:37.000Z",
        "notBefore": "2024-08-26T16:15:37.000Z",
        "serialNumber": "120064ECE1352507757C00A08B00020064ECE1",
        "thumbprint": "29063cd83537dbbd0a221705859b8f6e5bec5ad0",
        "signatureAlgorithm": "1.2.643.7.1.1.3.2",
        "signatureDigestAlgorithm": "1.2.643.7.1.1.2.2",
        "publicKeyAlgorithm": "1.2.643.7.1.1.1.1"
      },
```

```
      "certificateData": "MIIDEz...",
      "issuerName": "E=support@cryptopro.ru, C=RU, L=Moscow, O=CRYPTO-PRO LLC, CN=CRYPTO-PRO
Test Center 2",
      "serialNumber": "120064ECE1352507757C00A08B00020064ECE1",
      "signatureAlgorithm": "1.2.643.7.1.1.3.2",
      "signatureDigestAlgorithm": "1.2.643.7.1.1.2.2",
      "certs": [
        {
          "status": true,
          "subjectFriendlyName": "esia test3",
          "issuerFriendlyName": "CRYPTO-PRO Test Center 2",
          "subjectName": "CN=esia test3",
          "issuerName": "E=support@cryptopro.ru\r\nC=RU\r\nL=Moscow\r\nO=CRYPTO-PRO
LLC\r\nCN=CRYPTO-PRO Test Center 2",
          "notAfter": "2024-10-26T16:25:37.000Z",
          "notBefore": "2024-08-26T16:15:37.000Z",
          "serialNumber": "120064ECE1352507757C00A08B00020064ECE1",
          "thumbprint": "29063cd83537dbbd0a221705859b8f6e5bec5ad0",
          "signatureAlgorithm": "1.2.643.7.1.1.3.2",
          "signatureDigestAlgorithm": "1.2.643.7.1.1.2.2",
          "publicKeyAlgorithm": "1.2.643.7.1.1.1.1"
        }
      ],
      "isCertChainValid": true
    }
  ],
  "data": "JVBERi0xLjQKJdPr...=",
  "dataStamp": "JVBERi0xLjcKJ...=",
  "report": "JVBERi0xLjcKJ...="
}
```


Атрибуты ответа:

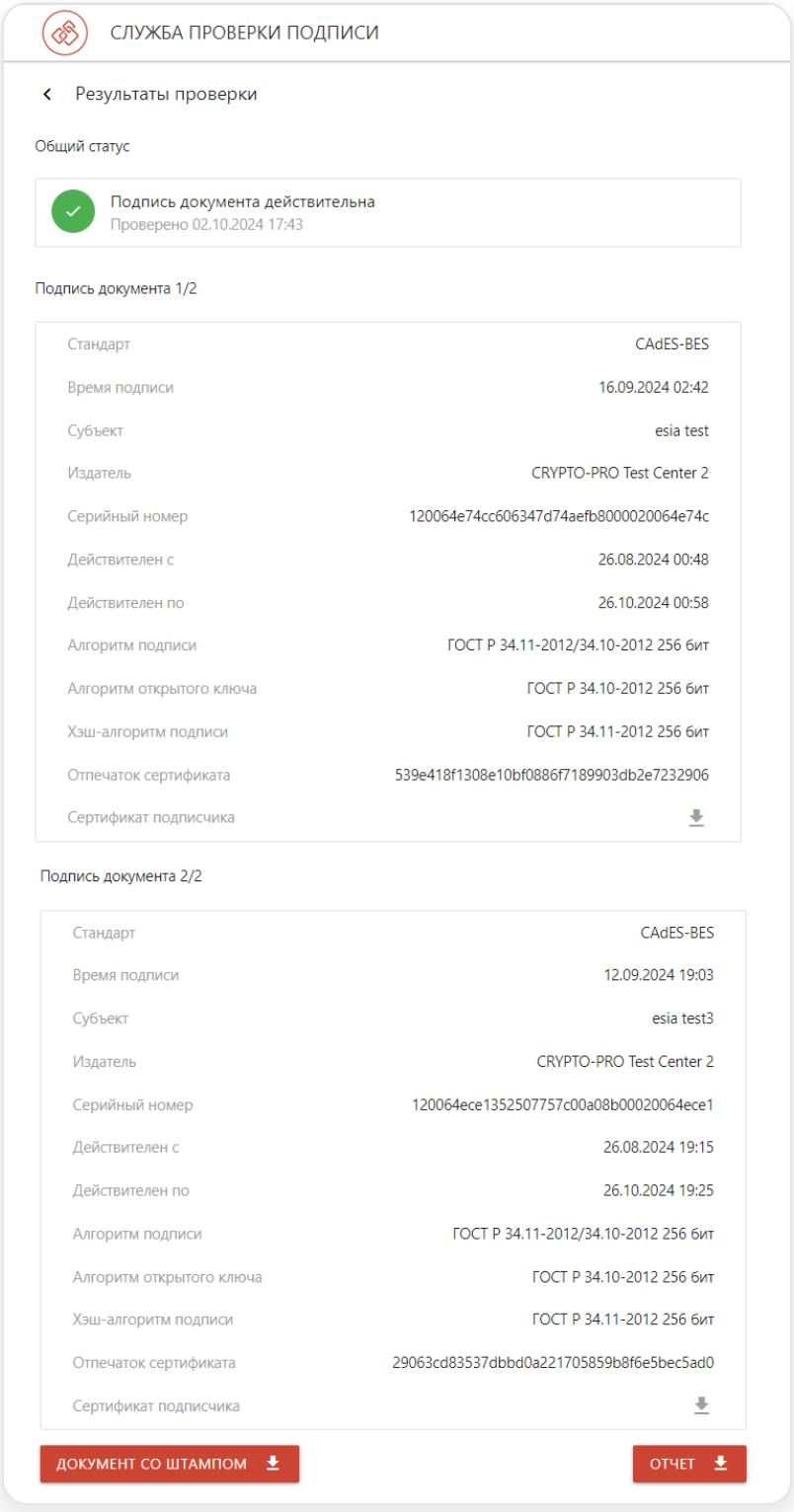
- **status**: Http статус
- **message**: Если произойдет ошибка, то причина ошибки будет указана здесь
- **isValid**: Общий результат проверки
- **isValidSign**: Результат математической корректности подписи
- **signs**: Информация о сертификатах
- **data**: Подписанные данные в формате Base64
- **dataStamp**: Подписанные данные с визуализацией штампов подписей в формате Base64 (для PDF документов)
- **report**: PDF-отчет проверки в формате Base64

Пример неуспешного ответа:

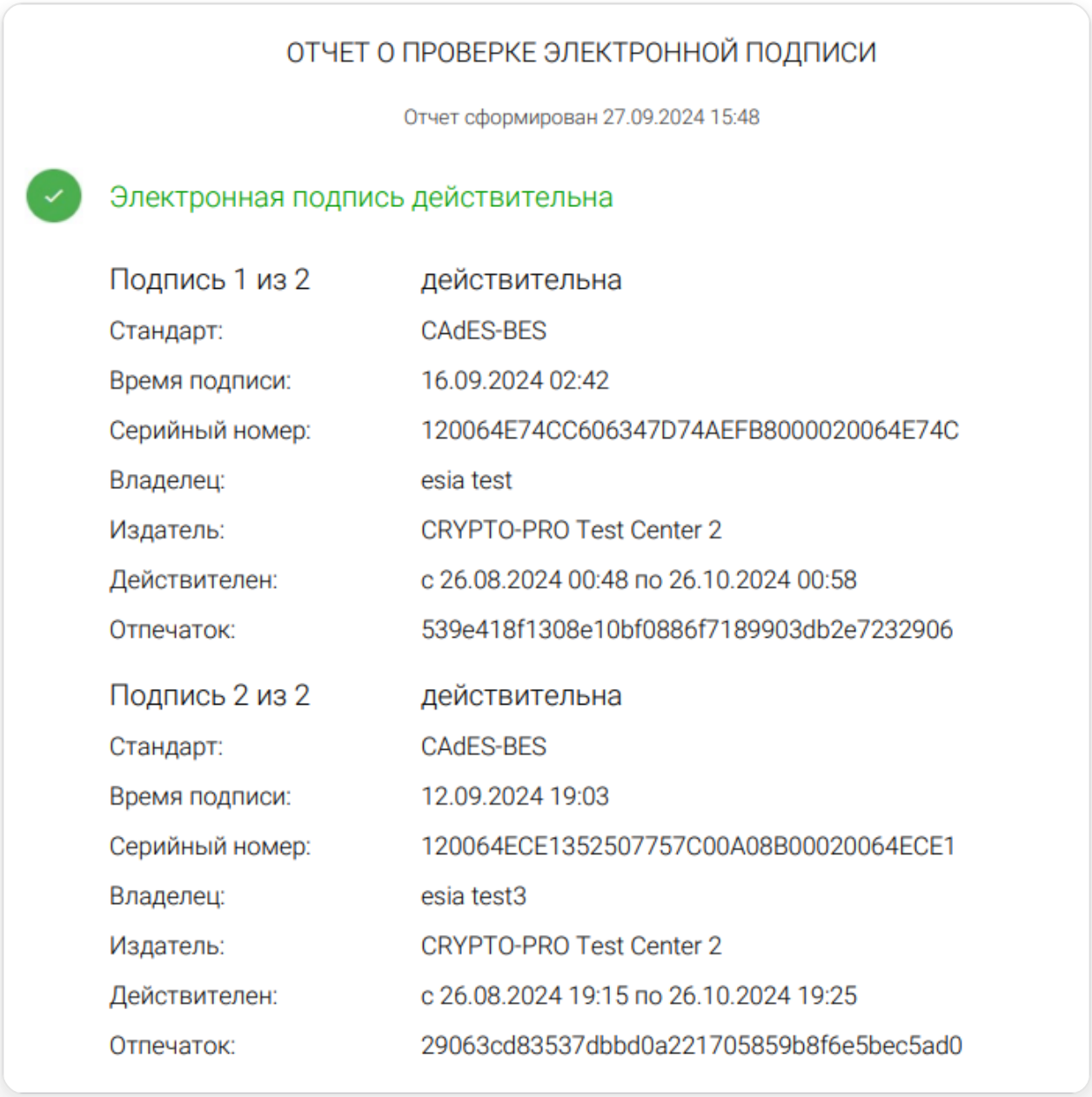

```
HTTP/1.1 400 Bad Request
Vary: Origin
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
Content-Length: 101
ETag: W/"65-ykHUL2wV5KLzsDntQL70C94heK8"
Date: Thu, 26 Sep 2024 12:42:04 GMT
Connection: keep-alive
Keep-Alive: timeout=5
{
    "message":
```

```
"Запрошена проверка прикрепленной CMS подписи, но подпись не содержит исходного документа",
    "error": "Bad Request",
    "statusCode": 400
}
```


Web-интерфейс сервиса:



Пример отчета о проверки подписи:



ОТЧЕТ О ПРОВЕРКЕ ЭЛЕКТРОННОЙ ПОДПИСИ

Отчет сформирован 27.09.2024 15:48

Электронная подпись действительна

Подпись 1 из 2	действительна
Стандарт:	CAdES-BES
Время подписи:	16.09.2024 02:42
Серийный номер:	120064E74CC606347D74AEFB8000020064E74C
Владелец:	esia test
Издатель:	CRYPTO-PRO Test Center 2
Действителен:	с 26.08.2024 00:48 по 26.10.2024 00:58
Отпечаток:	539e418f1308e10bf0886f7189903db2e7232906
Подпись 2 из 2	действительна
Стандарт:	CAdES-BES
Время подписи:	12.09.2024 19:03
Серийный номер:	120064ECE1352507757C00A08B00020064ECE1
Владелец:	esia test3
Издатель:	CRYPTO-PRO Test Center 2
Действителен:	с 26.08.2024 19:15 по 26.10.2024 19:25
Отпечаток:	29063cd83537dbbd0a221705859b8f6e5bec5ad0

Пример документа с визуализацией штампов подписей:


We do code reviews and may ask you to change things before we merge code.
This is a git-flow repo. We use the default git flow with a v version prefix. You shouldn't care much while writeing a feature but good to know.
Note that gitmoji is used in the commit messages. That's not a must but we think it's nice.
ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Открыть статью на сайте

Автоматизация проверки электронной подписи с использованием КриптоАРМ Сервер

Автоматизация проверки электронной подписи является одной из наиболее востребованных задач в ИТ-инфраструктуре, связанной с электронным документооборотом. С использованием КриптоАРМ Сервер и модуля trusted-crypto вы можете быстро интегрировать проверку электронной подписи в серверные приложения, обеспечивая надежную проверку подписанных документов в соответствии с законодательством.

В этой статье рассмотрим, как с помощью продукта КриптоАРМ Сервер и модуля trusted-crypto можно автоматизировать проверку электронной подписи.

КриптоАРМ Сервер и модуль trusted-crypto

КриптоАРМ Сервер — это продукт, состоящий из:

- КриптоАРМ SDK (набор модулей для Node.js), среди прочего обеспечивающий простую интеграцию проверки подписей;
- Полной документации для разработчиков;
- Примера сервиса для проверки подписей и визуализации штампов в PDF-документах.

Модуль trusted-crypto входит в состав КриптоАРМ SDK. В части проверки подписи поддержаны стандарты от CAdES-BES до CAdES-A, методы для построения цепочек и проверки сертификатов, работа с TSP и OCSP и многое другое.

Для работы модуля необходимо следующее окружение и лицензии:

- Node.js;
- КриптоПро CSP;
- Лицензии на КриптоАРМ и КриптоПро (серверные);
- Лицензии на КриптоПро TSP Client и КриптоПро OCSP Client, если нужна работа с усовершенствованной подписью.

Проверка электронной подписи на сервере

Процесс проверки электронной подписи с помощью модуля trusted-crypto включает несколько этапов: загрузка подписанного файла или импорт буфера, проверка подписи и сертификатов подписчиков.

Рассмотрим каждый этап подробнее на примере.

1. Загрузка подписанного файла

Первый шаг в процессе — это чтение подписанного файла. Для этого используем метод `load` или его асинхронный аналог `loadAsync`.

```
// Инициализация объекта для работы с подписанными данными
const cms = new trusted.cms.SignedData();

// Чтение файла с электронной подписью
cms.load("./outfile.sig");
```

Метод загружает файл с расширением `.sig`, содержащий электронную подпись. Возможна работа с разными форматами кодировками, такими как DER или BASE64.

Второй — импорт буфера:

```
//Метод определения кодировку для буфера
export const getBufferEncoding = (cmsBuffer: Buffer) => {
  const firstTwoSymbols = cmsBuffer.toString("utf8", 0, 2);
  if (firstTwoSymbols === "--" || firstTwoSymbols === "MI") {
    return "PEM";
  } else {
    return "DER";
  }
};

//Получим кодировку прочитав первые байты сообщения
const encoding = getBufferEncoding(cmsBuffer);

//Импорт буфера в объект подписи
cms.import(cmsBuffer, trusted.DataFormat[encoding!]);
```

2. Проверка подписи

После загрузки файла следующим шагом будет проверка подписи с помощью метода `verify`. Этот метод возвращает булево значение: `true` — если подпись действительна, и `false` — если возникла ошибка верификации.

```
//Используем метод проверки подписи
const res = cms.verify();
console.log(`Результат проверки: ${res}`);
```

Для асинхронной работы можно использовать метод `verifyAsync`.

```
cms.verifyAsync((error, result) => {
    if (error) {
        console.error(`Ошибка проверки подписи: ${error}`);
    } else {
        console.log(`Результат проверки: ${result}`);
    }
});
```

3. Работа с отсоединенной подписью

Отсоединенная подпись (detached signature) — это подпись, которая передается отдельно от подписываемого контента. Для проверки такой подписи необходимо загрузить как файл с подписью, так и сам подписанный документ.

```
cms.load("./outfile.sig");
cms.content = {
    type: trusted.cms.SignedDataContentType.url,
    data: "./data.docx"
};
const result = cms.verify();
console.log(`Откреплennая подпись действительна: ${result}`);
```

4. Проверка конкретного подписчика

Если необходимо проверить подпись конкретного подписчика (соподпись), можно получить коллекцию подписчиков с помощью метода `signers` и затем передать конкретного подписчика в метод `verify`.

```
const signers = cms.signers();
const signer = signers.items(0); // Выбираем первого подписчика
const result = cms.verify(signer);
console.log(`Подпись подписчика действительна: ${result}`);
```

5. Проверка сертификатов подписчика

Для проверки сертификата подписчика можно извлечь его из объекта Signer или передать сертификат отдельно. Далее строим цепочку и проверяем сертификат. Для построения цепочки используется хранилище КриптоПро. При этом списки отзыва и промежуточные сертификаты будут скачаны автоматически (если не установлены в хранилище и есть доступ к сети).

```
function verifyCertificate(cert) {
  return new Promise((resolve, reject) => {
    trusted.utils.Csp.verifyCertificateChainAsync(cert, (errorMsg, result) => {
        if (errorMsg) {
                return reject({
                    title: 'Не удалось проверить цепочку сертификатов для ' + cert?.subjectFri
endlyName,
                    text: 'verifyCertificateChainAsync: ' + errorMsg,
                    timestamp: new Date().toISOString(),
                });
        }

        resolve(result);
    });
  });
}


function buildChain(cert) {
  return new Promise((resolve, reject) => {
    trusted.utils.Csp.buildChainAsync(cert, (errorMsg, certCollection) => {
        if (errorMsg) {
                return reject({
```

```
                    title: 'Не удалось построить цепочку сертификатов для ' + cert?.issuerFrie
ndlyName,
                    text: 'buildChainAsync: ' + errorMsg,
                    timestamp: new Date().toISOString(),
                });
        }

        resolve(certCollection);
    });
  });
}


const certificate = signer.certificate;
const certChain = await buildChain(certificate);

for (let i = 0; i < certChain?.length; i++) {
  const cert = certChain?.items(i);
  const result = await verifyCertificate(cert);
  console.log(`Статус сертификата: ${result}`);
}
```

Использование в микросервисной архитектуре

В микросервисной архитектуре проверка электронной подписи с помощью КриптоАРМ Сервер и trusted-crypto может быть вынесена в отдельный сервис, который будет отвечать исключительно за криптографические операции. Такой подход упрощает масштабирование системы и повышает безопасность, так как каждый микросервис имеет ограниченный доступ к различным компонентам системы.

Пример возможного REST API:

Проверка подписи документа

Метод: `POST /api/v1/signature/verify`

Описание: Проверяет электронную подпись документа.

Запрос:

- Content-Type: `application/json` или `multipart/form-data` (если файлы передаются через форму).

Параметры:

- `document` (обязательный, `Base64`, `File`): Документ, для которого нужно проверить подпись.
- `signature` (обязательный, `Base64`, `File`): Подпись в формате CMS.
- `certificate` (опциональный, `Base64`, `File`): Сертификат, используемый для проверки подписи (если не встроен в подпись).
- `type` (опциональный, `string`): Тип подписи (`CAdES`, `CMS`, `XLT1`). По умолчанию — автоопределение.
- `detached` (опциональный, `boolean`): Указывает, является ли подпись откреплённой от документа (detached).

Ответ:

- Status 200: Подпись валидна.

```
{
  "status": "valid",
  "details": {
  "issuer": "Issuer CN",
  "subject": "Subject CN",
  "valid_from": "2024-01-01T00:00:00Z",
  "valid_to": "2026-01-01T00:00:00Z",
  "signing_time": "2024-10-14T12:30:00Z"
  }
}
```

- Status 400: Ошибка валидации подписи.

```
{
  "status": "invalid",
  "error": "Signature does not match the document"
}
```

Примеры кода и документация

Для более глубокого изучения возможностей модуля trusted-crypto и его применения для автоматической проверки электронной подписи в форматах CAdES-BES, CAdES-T, CAdES-X Long Type 1 и CAdES-A на сервере рекомендуем ознакомиться с документацией и примерами кода. Они помогут вам интегрировать проверку подписей в серверные и микросервисные архитектуры.

Документация на модуль trusted-crypto: trusted-crypto.

Попробуйте бесплатную демонстрацию возможностей КриптоАРМ Сервер и модуля trusted-crypto, загрузив SDK.

Запросите триальную лицензию заполнив форму на сайте cryptoarm.ru.

Если у вас есть вопросы по интеграции модуля trusted-crypto в ваши серверные решения или вы хотите узнать больше о возможностях автоматической проверки электронной подписи, свяжитесь с нами по электронной почте sales@cryptoarm.ru. Мы также готовы предложить услуги по интеграции решений «под ключ».

Открыть статью на сайте

Автоматическое подписание обезличенным сертификатом

В этой статье мы рассмотрим, что такое обезличенная электронная подпись, как она регулируется законодательством и каким образом можно автоматизировать процессы подписания с помощью таких инструментов, как КриптоАРМ Сервер (КриптоАРМ SDK).

Процесс автоматической проверки подписи описан в нашей статье «Автоматизация проверки электронной подписи с использованием КриптоАРМ Сервер».

Что такое обезличенная электронная подпись и где получить сертификат

Обезличенная электронная подпись представляет собой подпись, использующую сертификат без указания персональных данных владельца, и предназначена для юридических лиц и индивидуальных предпринимателей (в сертификате для ИП все же указываются данные владельца). Она выполняет функцию, схожую с печатью организации, что позволяет автоматизировать процессы подписания в информационных системах. Сертификат, используемый для такой подписи, не закрепляется за конкретным сотрудником.

Федеральная налоговая служба России выпускает обезличенные сертификаты, которые применяются для автоматического создания и проверки электронной подписи в электронных документах.

Центробанк предоставляет квалифицированные обезличенные сертификаты для участников, ведущих деятельность в соответствии с ч. 1 статьи 76.1 Закона о Центральном банке.

Федеральное казначейство обеспечивает выпуск обезличенных сертификатов для нужд органов власти.

Применение обезличенной электронной подписи согласно Федеральному закону № 63-ФЗ «Об электронной подписи»

Область применения обезличенной подписи ограничена: её нельзя использовать в документах, которые направляются вне информационных систем или для участия в государственных закупках. Применение обезличенной электронной подписи разрешено в автоматизированных информационных системах — как государственных (например СМЭВ), так и частных, если это предусмотрено федеральным законодательством.

В соответствии с Законом № 63-ФЗ «Об электронной подписи» такие подписи предназначены для автоматической подписи и проверки электронных документов в системах, где не требуется участие человека. В этом случае назначается ответственное лицо за процесс создания и проверки подписи, а при его отсутствии ответственность возлагается на руководителя организации.

Преимущества автоматического подписания с использованием обезличенного сертификата

Обезличенная электронная подпись позволяет компаниям значительно улучшить процессы обработки документов:

1. Ускорение процессов. Подпись создается автоматически, что исключает задержки, связанные с ручным подписанием.
2. Исключение человеческого фактора. Автоматизация снижает риск ошибок, которые могут возникнуть при ручной работе.
3. Соответствие законодательству. Использование в автоматических системах соответствует всем законодательным требованиям.
4. Легкость интеграции. Обезличенная подпись интегрируется в информационные системы, а КриптоАРМ SDK содержит все необходимые методы для реализации.

Автоматическое подписание с использованием КриптоАРМ Сервер

Для интеграции обезличенной электронной подписи можно использовать КриптоАРМ Сервер и модуль trusted-crypto. С помощью этого модуля можно реализовать автоматическое создание и проверку подписей в информационных системах.

Рассмотрим несколько этапов работы с электронными подписями.

Пример кода для автоматического подписания

1. Чтение обезличенного сертификата

Загрузка файла сертификата осуществляется с помощью метода `load` или асинхронного `loadAsync`.

```
const certFile = "mycertificate.cer";
const cert = trusted.pki.Certificate.load(DEFAULT_RESOURCES_PATH + "/" + certFile, trusted.Dat
aFormat.DER);
```

Этот метод загружает сертификат, который может быть закодирован в формате DER или BASE64. Сертификат должен быть установлен в хранилище сертификатов с привязкой к закрытому ключу, в момент подписи trusted-crypto получит контекст ключа.

Альтернативный вариант работы с сертификатами — работа с классом хранилища `trusted.pkistore.PkiStore`.

2. Подпись данных или файла

Подпись выполняется методом `sign`. После подписания данные можно экспортировать в буфер или записать в файл.

```
var signer;
var policies;

sd = new trusted.cms.SignedData();

sd.policies = ["detached"];

   sd.content = {
         type: trusted.cms.SignedDataContentType.buffer,
         data: "Hello world"
   };

sd.sign(cert);

sd.export();
```

Для асинхронной подписи можно использовать метод `signAsync`.

```
function (done) {
                 cms = new trusted.cms.SignedData();

                 cms.policies = isDetached ? ["detached"] : [];

                 cms.content = {
                     type: trusted.cms.SignedDataContentType.url,
                     data: DEFAULT_RESOURCES_PATH + "/" + plainFile
                 };
                 cms.signAsync(cert, done);
             })
```

3. Добавление подписи (соподпись)

```
 const csd = new trusted.cms.SignedData();

 csd.import(signBuffer)

 const cert = trusted.pki.Certificate.load(DEFAULT_RESOURCES_PATH + "/cert-with-pin.cer", trus
ted.DataFormat.DER);

csd.sign(cert);
```

4. Усовершенствованная подпись

```
var sdCades;
            var connSettings = new trusted.utils.ConnectionSettings();
            connSettings.Address = TSP_SERVICE_ADDRESS;
```

```
            sdCades = new trusted.cms.SignedData();

            sdCades.content = {
                type: trusted.cms.SignedDataContentType.buffer,
                data: "CAdES-" + cadesType + " test 1",
            };

            var cadesParams = new trusted.cms.CadesParams();
            cadesParams.cadesType = cadesTypeByName(
                cadesType,
                "Unknown type " + cadesType
            );
            cadesParams.connSettings = connSettings;
            cadesParams.tspHashAlg = "1.2.643.7.1.1.2.2";
            sdCades.signParams = cadesParams;

            assert.doesNotThrow(function () {
                sdCades.sign(cert);
            });

            assert.notStrictEqual(
                sdCades.export(),
                undefined,
                "Unable to export CAdES"
            );
            sdCades.save(
                DEFAULT_OUT_PATH + "/" + signCades1,
                trusted.DataFormat.PEM
            );
```

Для более подробного ознакомления с функционалом модуля `trusted-crypto` можно обратиться к нашей документации, доступной по ссылке.

Заключение

Использование обезличенной электронной подписи позволяет ускорить обработку данных, снизить затраты на администрирование и обеспечить соответствие требованиям законодательства. Использование КриптоАРМ Сервер предоставляет организациям инструменты для автоматизации подписания и проверки данных.

По вопросам реализации сервиса автоматического подписания обращайтесь на наш email sales@cryptoarm.ru.

Открыть статью на сайте

КриптоАРМ. Возможности по интеграции и автоматизации

Что такое КриптоАРМ API и SDK? Отличия и возможности

КриптоАРМ предлагает два инструмента для интеграции и автоматизации: API и SDK. Оба варианта упрощают автоматизацию криптографических операций, но имеют различную сферу применения.

КриптоАРМ API — это программный интерфейс, который позволяет интегрировать КриптоАРМ с другими информационными системами или приложениями. С его помощью можно вызывать функции КриптоАРМ с заданными параметрами. API доступно в двух вариантах: через веб-протокол (похожий на JSON-RPC 2.0) и через локальное API для вызовов из консольных приложений. КриптоАРМ устанавливается на клиентском устройстве, и пользователь подтверждает операции.

Основное преимущество использования API — отсутствие необходимости разработки кода для криптографических операций. Вы просто передаете параметры операции и получаете результаты через контроллер. API работает на всех операционных системах, включая мобильные устройства.

Пример использования КриптоАРМ API: Ваш портал выставляет счета, и требуется организовать юридически значимый электронный документооборот (ЭДО). С помощью API можно добавить кнопку «Подписать» в личный кабинет контрагента. КриптоАРМ API запустит мастер подписи с уже подготовленными документами, и пользователю останется только подписать их. Документы автоматически отправятся обратно в вашу систему.

Для 1С-Битрикс есть готовый бесплатный модуль: КриптоАРМ Документы.

КриптоАРМ SDK — это набор модулей для разработки нового сервиса или продукта. Чаще всего используется на сервере. SDK не содержит пользовательского интерфейса (UI) и требует написания скриптов на TypeScript/JavaScript. SDK идеально подходит для создания автоматизированных решений, где требуется обезличенная подпись документов или их проверка.

Пример использования КриптоАРМ SDK: Ваш интернет-магазин должен автоматически подписывать счета. С помощью SDK вы можете написать код для автоматической подписи, используя простые верхнеуровневые методы. Это избавляет вас от необходимости разбираться в сложных криптографических стандартах.

Функционал КриптоАРМ API и SDK: Что доступно?

Через **КриптоАРМ API** можно выполнить любую операцию, которые уже доступны в графическом интерфейсе КриптоАРМ.

Операции:

- подписать один или несколько файлов;
- создать копию с визуальным штампом;
- встроить подпись в PDF-документ;
- проверить действительность электронной подписи одного или нескольких файлов;
- зашифровать файлы;
- расшифровать файлы;
- создать запрос на получение сертификата;
- сгенерировать ключевую пару;
- экспортировать сертификат или сертификаты;
- импортировать сертификат или сертификаты;
- показать информацию о сертификате;
- выполнить диагностику рабочего места;
- сформировать новое письмо.

КриптоАРМ SDK состоит из модулей для Node.js:

- trusted-crypto — подпись, проверка, шифрование и расшифрование, работа с сертификатами;
- trusted-curl — защищенное сидение с поддержкой ГОСТ TLS;
- trusted-mailcore2 — сбор и разбор писем, протоколы IMAP и SMTP;
- trusted-pdf — преобразование PDF в PDF-A, сертификация PDF с разметкой областей для оттисков подписи, подпись и проверка подписи встроенной в PDF.

В каких версиях КриптоАРМ есть API или SDK?

Продукт	API	SDK
КриптоАРМ ГОСТ 2.2.4 (сертифицирован)	Нет	Нет
КриптоАРМ ГОСТ 2.5 (desktop)	Да	Да
КриптоАРМ 2 (mobile)	Да	Нет
КриптоАРМ ГОСТ 3 (desktop) (сертифицирован)	Да (+ локальное API)	Да
КриптоАРМ ГОСТ 3 (mobile) (сертифицирован)	Да	Нет
КриптоАРМ 6 (desktop)	Да (+ локальное API)	Да

Продукт	API	SDK
КриптоАРМ 1.0 Mobile	Да	Нет
КриптоАРМ 1.0 Mobile для Aurora OS	Нет	Нет

Требуется ли оценка влияния?

Для использования КриптоАРМ в некоторых сценариях может потребоваться проведение исследований по оценке влияния (СФ).

В КриптоАРМ ГОСТ 3 возможны следующие варианты применения:

1. Использование графических интерфейсов СКЗИ.

   В данном варианте использования СКЗИ дополнительных исследований **не требуется**.

2. Программный вызов **КриптоАРМ API**.

   В данном варианте использования СКЗИ требуется проведение исследований **по оценке влияния** СФ на СКЗИ.

3. Использование **КриптоАРМ SDK** рассматривается как встраивание СКЗИ во вновь разрабатываемое или существующее прикладное программное обеспечение, программно-аппаратные и аппаратные решения. В данном случае требуется **проведение тематических исследований** СФ со встроенным СКЗИ как самостоятельного шифровального (криптографического) средства.

Если вам требуется помощь в оценке влияния, оставьте заявку на sales@cryptoarm.ru.

Какие лицензии на продукты нужны?

Для использования КриптоАРМ SDK необходима серверная лицензия. КриптоАРМ API может работать как с клиентскими, так и серверными лицензиями на КриптоАРМ.

Для API будет доступен дополнительный вариант лицензирования — транзакционный. В этом случае оператор ИС приобретает пакет лицензий и распространяет своим клиентам клиентские лицензии вместе с параметрами операции. Пользователю не надо покупать КриптоАРМ или вручную вводить лицензии.

Так же требуется действующая лицензия на криптопровайдер КриптоПро CSP.

Где посмотреть документацию и примеры для API или SDK?

Ознакомьтесь с документацией КриптоАРМ API и КриптоАРМ SDK.

Пример по вызову API реализован на этом стенде, его исходный код - ссылка.

В качестве примера использования КриптоАРМ SDK мы реализовали сервис по проверке подписи и постановке визуальных штампов. Сервис доступен по ссылке, его исходный код открыт - ссылка.

Мы готовы помочь вам с интеграцией API и SDK в вашу систему, по всем вопросам обращайтесь на sales@cryptoarm.ru.

Открыть статью на сайте

Интеграция КриптоАРМ во внешнюю информационную систему

В статье "КриптоАРМ. Возможности по интеграции и автоматизации" мы рассмотрели основные возможности работы КриптоАРМ через API и SDK. В этом материале мы углубимся в использование **КриптоАРМ API**, предложив примеры и сценарии интеграции в информационные системы.

КриптоАРМ API. Возможности и преимущества

КриптоАРМ API позволяет выполнять такие операции, как: электронная подпись документов, шифрование и расшифрование данных, проверка электронных подписей, а также управление сертификатами. Кроме того, доступны возможности для автоматизации генерации запросов на сертификаты и диагностики рабочего места пользователя. Это делает КриптоАРМ API универсальным решением для обеспечения безопасности и организации юридически значимого документооборота, повышая эффективность рабочих процессов.

Основные преимущества использования API:

- **Быстрая интеграция криптографии**. Не требуется разрабатывать сложные криптографические модули, достаточно передать параметры, а результат операции будет готов через несколько кликов.
- **Кроссплатформенная поддержка**. API доступен как на настольных системах (Windows, macOS, Linux), так и на мобильных платформах (iOS, Android), что делает его универсальным инструментом для любого приложения.
- **Безопасность данных**. Все взаимодействия с API осуществляются по защищенным каналам связи с использованием TLS, что гарантирует сохранность передаваемых данных.
- **Простота использования**. Для взаимодействия с КриптоАРМ API используется протокол cryptoarm://, который легко интегрируется в существующие решения.

Примеры использования КриптоАРМ API

Рассмотрим, как API КриптоАРМ может быть применен в реальных сценариях.

1. Электронный документооборот: подпись документов

Представьте, что ваше веб-приложение позволяет пользователям обмениваться юридически значимыми документами. В этом случае можно использовать команду signAndEncrypt, чтобы быстро и безопасно подписывать документы, вызывая операцию из браузера пользователя.

Как это работает:

1. Пользователь заходит на веб-портал.

2. Выбирает нужные файлы и действие, например, «Подписать».
3. Портал генерирует ссылку с протоколом `cryptoarm://`, которую пользователь кликает для начала операции.
4. Если КриптоАРМ не запущен, программа автоматически стартует и получает параметры от сервера в виде JSON.
5. КриптоАРМ выполняет операцию (подписание и шифрование) и возвращает результат на сервер.

Пример ссылки для вызова КриптоАРМ:

```
cryptoarm://signAndEncrypt/https://yourdomain.com/api/sign?id=12345
```

```
{
  "jsonrpc": "2.0",
  "result": {
    "operation": [
      "SIGN"
    ],
    "props": {
      "license": "",
      "files": [
        {
          "name": "file1.txt",
          "url": "https://yourdomain.com/files/file1.txt",
          "id": 1,
          "urlDetached": ""
        }
      ],
      "extra": {
        "token": "9c7101f7-9c47-4481-b4da-a6a497abde08",
        "signType": 1,
        "signStandart": 0,
        "signEncoding": 0,
        "signatureExtension": "sgn"
      },
      "uploader": "https://yourdomain.com/api/upload"
    }
  },
  "id": "12345"
}
```

Этот запрос откроет мастер подписи и шифрования на основе переданных параметров. После завершения операции результат отправляется обратно на сервер. Пример JSON-ответа после успешной операции:

```
{

    "jsonrpc":"2.0",
    "method":"signAndEncrypt.outDirectResults",
    "params":{
        "id":"3a2fca10-fb2c-45eb-a61e-2e0da2a92bf0",
        "status":"Completed",
        "directResults":[
            {
                "id":1,
```

```
                "signValid":true,
                "signers":[
                    {
                        "isValid":true,
                        "signingTime":1730531671000,
                        "signerCertificate":{
                            "hash":"f9cfecf5e1e986d1aa227538469e2b4c34e51a4b",
                            "issuerFriendlyName":"test",
                            "issuerName":"C=RU\r\nCN=test",
                            "subjectFriendlyName":"test",
                            "subjectName":"C=RU\r\nCN=test",
                            "status":true,
                            "serial":"049042780942464741",
                            "notAfter":2040208687000,
                            "notBefore":1724675887000,
                            "rootCAMinComSvyaz":false
                        }
                    }
                ],
                "out": "MIAGCSqGSIb3DQEHAqCAMIACAQ....."
                "outSignStamp":null
            }
        ]
    }
}
```


Этот сценарий может использоваться для электронной подписи договоров и счетов, обеспечивая юридическую силу документов.

2. Шифрование файла

Для команды `signAndEncrypt` доступна операция шифрования (может использоваться одновременно с подписью, тогда файл подпишется, а потом будет зашифрован). Вы можете передать для шифрования сертификаты получателей или оставить эти поля незаполненными, тогда клиент сам сможет указать нужных получателей.


```
{
    "jsonrpc": "2.0",
    "result": {
      "operation": [
        "ENCRYPT"
      ],
      "props": {
        "license": "",
        "files": [
          {
            "name": "file1.txt",
            "url": "https://yourdomain.com/public/files/file1.txt",
            "id": 1,
            "urlDetached": ""
          }
        ],
        "extra": {
          "token": "9c7101f7-9c47-4481-b4da-a6a497abde08",
          "encryptEncoding": 1,
          "encryptAlgorithm": 1,
          "encryptCertificates": [
```

```
            "MIICFTCCAcKgAwIBAgIJQCFliUQmiIAQMA…….",
            "MIICZDCCAdCgAwIBAgIIkwgSETAwKEIwCgYIK……"
            ]
        },
        "uploader": "https://yourdomain.com/upload"
      }
    },
    "id": "18d6dc4d-27b5-4558-b6b2-e7a97d0c99d9"
  }
```

Пример JSON-ответа после успешной операции шифрования:

```
{
    "jsonrpc":"2.0",
    "method":"signAndEncrypt.outDirectResults",
    "params":{
        "id":"18d6dc4d-27b5-4558-b6b2-e7a97d0c99d9",
        "status":"Completed",
        "directResults":[
            {
                "id":1,
                "out": "MIAGCSqGSIb3DQEHA6CAMIA...."
                "outSignStamp":null
            }
        ]
    }
}
```

3. Формирование письма с вложениями

Если ваша информационная система предполагает обмен файлами (например с контрагентами), вы можете использовать команды КриптоАРМ API `sendMail`. ИС может передать все нужные параметры письма (получатели, тело письма, вложения, опции подписи и шифрования), а клиенту останется подтвердить отправку письма в один клик.

Пример параметров формирования письма из API:

```
{
    "jsonrpc": "2.0",
    "result": {
        "operation": "SEND",
        "props": {
            "mailProps": {
                "to": [
                    "test0@yandex.ru"
                ],
                "cc": [
                    "test1@gmail.com",
                    "test2@icloud.com"
                ],
                "bcc": [
                    "test@yandex.ru"
                ],
                "subject": "Message subject",
                "content": "text message content",
                "htmlContent": "<div>HTML message content</div>",
```

```
                "attachments": [
                    {
                        "name": "file1.txt",
                        "url": "https://yourdomain.com/public/files/file1.txt",
                        "id": 1,
                        "urlDetached": ""
                    },
                    {
                        "name": "file2.txt",
                        "url": "https://yourdomain.com/public/files/file2.txt",
                        "id": 2,
                        "urlDetached": ""
                    },
                    {
                        "name": "file4.pdf",
                        "url": "https://yourdomain.com/public/files/file4.pdf",
                        "id": 4,
                        "urlDetached": ""
                    }
                ],
                "sign": true,
                "encrypt": true,
                "deliveryReport": false,
                "readReport": false
            }
        }
    },
    "id": "fceff54e-c4a4-4282-80db-2a581fe7161b"
}
```


Заключение

Интеграция КриптоАРМ API в информационные системы — это простой и эффективный способ реализовать электронный документооборот, ускорить процессы и упростить работу с электронными подписями. Подробная документация по ссылке, а примеры запросов в нашем публичном репозитории.

Хотите узнать больше о возможностях интеграции КриптоАРМ в ваши информационные системы? Свяжитесь с нами по sales@cryptoarm.ru для бесплатной консультации.

Открыть статью на сайте

Сервис проверки и визуализации электронной подписи

В статье «Автоматизация проверки электронной подписи с использованием КриптоАРМ Сервер» мы описали, как с помощью КриптоАРМ SDK реализовать проверку подписи. В этой статье покажем, как с помощью REST API можно проверить электронную подпись и визуализировать штампы на PDF-документах. Попробовать сервис можно здесь, а исходный код доступен в репозитории.

Обзор сервиса и его компонентов

Сервис проверки и визуализации электронной подписи предлагает следующие возможности:

1. **REST API** для обработки документов и проверки подписей.
2. **Визуализацию штампов** электронной подписи для PDF-документов.
3. **Создание отчетов о проверке подписей** для документов других форматов.

Наш тестовый сервис позволяет загружать документы и подписи в различных форматах (поддерживается проверка подписей PADES для PDF). Для PDF-файлов есть возможность создавать копию с визуальным штампом, а для других форматов — формировать отчет.

Сервис использует следующие библиотеки из состава КриптоАРМ SDK:

- **trusted-crypto** — для проверки подписей и сертификатов;
- **trusted-pdf** — для работы с PDF.

Зачем нужна визуализация электронной подписи?

Электронная подпись подтверждает авторство, неизменность и целостность данных. Однако бывает необходимо (например при для пользователей, работающих с бумажными копиями электронных документов), чтобы документы содержали отметку, визуально подтверждающую факт подписания. Визуализация электронной подписи — это добавление специального штампа на электронный документ, состоящего из:

- фразы «Документ подписан электронной подписью»;
- номера сертификата ключа подписи;
- ФИО владельца сертификата;
- срока действия сертификата;
- дополнительных атрибутов, такие как эмблема или логотип (при необходимости).

Согласно п. 5.23 ГОСТ Р 7.0.97-2016 визуализация подписи должна быть аналогична месту подписи в бумажном документе, а также видимой и читаемой.

Примеры запросов к сервису

1. Загрузка и проверка подписанного документа

Первый шаг — загрузка документа и подписи с помощью метода `POST /api/v1/signature/verify`. Этот метод проверяет действительность электронной подписи.

Запрос:

- Метод: `POST /cms/verify`
- Заголовок: `Content-Type: multipart/form-data`
- Параметры:
- `data` (`Base64`): документ, для которого нужно проверить подпись, если сама подпись откреплennая.
- `cms` (обязательный, `Base64``): CMS подпись.

Пример ответа:


```
{
  "status": 200,
  "message": "",
  "isValid": true,
  "isValidSign": true,
  "signs": [
    {
    "cadesTypeName": "CAdES-BES",
    "certificate": {
        "version": 3,
        "status": true,
        "subjectFriendlyName": "esia test3",
        "issuerFriendlyName": "CRYPTO-PRO Test Center 2",
        "subjectName": "CN=esia test3",
        "issuerName": "E=support@cryptopro.ru\r\nC=RU\r\nL=Moscow\r\nO=CRYPTO-PRO
LLC\r\nCN=CRYPTO-PRO Test Center 2",
        "notAfter": "2024-10-26T16:25:37.000Z",
        "notBefore": "2024-08-26T16:15:37.000Z",
        "serialNumber": "120064ECE1352507757C00A08B00020064ECE1",
        "thumbprint": "29063cd83537dbbd0a221705859b8f6e5bec5ad0",
        "signatureAlgorithm": "1.2.643.7.1.1.3.2",
        "signatureDigestAlgorithm": "1.2.643.7.1.1.2.2",
        "publicKeyAlgorithm": "1.2.643.7.1.1.1.1"
    },
    "certificateData": "MIIDEz...",
    "issuerName": "E=support@cryptopro.ru, C=RU, L=Moscow, O=CRYPTO-PRO LLC, CN=CRYPTO-PRO
Test Center 2",
    "serialNumber": "120064ECE1352507757C00A08B00020064ECE1",
    "signatureAlgorithm": "1.2.643.7.1.1.3.2",
    "signatureDigestAlgorithm": "1.2.643.7.1.1.2.2",
    "certs": [
        {
        "status": true,
        "subjectFriendlyName": "esia test3",
        "issuerFriendlyName": "CRYPTO-PRO Test Center 2",
        "subjectName": "CN=esia test3",
        "issuerName": "E=support@cryptopro.ru\r\nC=RU\r\nL=Moscow\r\nO=CRYPTO-PRO
LLC\r\nCN=CRYPTO-PRO Test Center 2",
```

```
        "notAfter": "2024-10-26T16:25:37.000Z",
        "notBefore": "2024-08-26T16:15:37.000Z",
        "serialNumber": "120064ECE1352507757C00A08B00020064ECE1",
        "thumbprint": "29063cd83537dbbd0a221705859b8f6e5bec5ad0",
        "signatureAlgorithm": "1.2.643.7.1.1.3.2",
        "signatureDigestAlgorithm": "1.2.643.7.1.1.2.2",
        "publicKeyAlgorithm": "1.2.643.7.1.1.1.1"
        }
    ],
    "isCertChainValid": true
    }
  ]
}
```


2. Визуализация штампа электронной подписи на PDF-документах

Для PDF-документов сервис позволяет добавить визуальный штамп, отображающий основные сведения о подписанте. Такой штамп делает документ наглядным и удобным для работы в бумажном виде.

3. Генерация отчета о проверке

Когда сервис получает документ в формате, отличном от PDF, вместо визуализации штампа можно запросить отчет о проверке подписи.

Заключение

Итак, мы рассмотрели наш пример сервиса проверки и визуализации электронной подписи с REST API. Визуализация штампа упрощает работу с PDF-документами, а отчеты о проверке позволяют легко получить подтверждение подлинности для других типов документов.

Попробуйте бесплатную демонстрацию сервиса.

Используя исходный код тестового стенда вы можете реализовать свое решение. Мы всегда рады помочь с реализацией и расширить функционал, вопросы можно отправлять нам на почту sales@cryptoarm.ru.

Открыть статью на сайте

Работа с почтой в Node.js. Примеры и возможности КриптоАРМ Сервер

Интеграция электронной почты — одна из важных функций для серверных приложений, будь то отправка уведомлений, получение сообщений или автоматизация рассылки файлов. Если вы разрабатываете серверные приложения на платформе **Node.js** и вам требуется безопасная работа с почтовыми протоколами IMAP и SMTP, **КриптоАРМ Сервер** предлагает удобное решение — модуль **trusted-mailcore2**.

В этой статье мы рассмотрим основные возможности модуля **trusted-mailcore2**, его уникальные преимущества и продемонстрируем примеры кода, которые помогут вам быстро внедрить его в ваш проект.

Основные возможности trusted-mailcore2

Модуль **trusted-mailcore2** создан для решения всех ключевых задач работы с почтой в Node.js. Он построен на основе opensource библиотеки MailCore2, но с важными доработками в части безопасности — нами добавлена поддержка российских криптографических алгоритмов.

Расширения безопасности:

- **Криптографическая защита писем**: Поддержка подписания и шифрования через модуль trusted-crypto, который обеспечивает защиту ГОСТ алгоритмами.
- **Надёжные TLS соединения**: Благодаря интеграции с **cpcurl** от КриптоПро обеспечена поддержка ГОСТ TLS для безопасных соединений с почтовыми серверами.
- **Работа с сертифицированным криптопровайдером КриптоПро CSP.**

Эти улучшения делают **trusted-mailcore2** хорошим решением для проектов, требующих высокого уровня безопасности в работе с электронной почтой.

Примеры использования модуля

trusted-mailcore2 поддерживает все основные почтовые протоколы, что делает его универсальным инструментом для работы с почтовыми серверами. Вы можете работать с IMAP, SMTP и POP3 в зависимости от потребностей вашего проекта. Классы по работе с письмом (MessageBuilder, MessageHeader, MessageParser) и вложениями (Attachment) позволяют реализовать любые задачи по формированию и чтению писем, например рассылку файлов по адресатам в автоматическом режиме.

Рассмотрим примеры кода для некоторых задач.

1. Отправка электронных писем (SMTP)

С помощью **trusted-mailcore2** можно легко отправлять письма через протокол SMTP. Доступны отправка писем с вложениями, использование разных форматов (текстовый или HTML) и настройка параметров отправки.

```
//Создаем объект сессии и зададим параметры подключения
const session = new SMTPAsyncSession();
session.hostname = "smtp-mail.outlook.com";
session.port = 587;
session.username = "test@hotmail.com";
session.password = "testpassword";
session.authType = AuthType.AuthTypeSASLLogin;
session.connectionType = ConnectionType.ConnectionTypeStartTLS;

//Создаем сообщение. Указываем заголовки и формируем тело
const createMessage = (): Buffer => {
    const testHeader = new MessageHeader();
    testHeader.from = "test@hotmail.com";
    testHeader.to = ["test1@hotmail.com", "test2@hotmail.com"];
    testHeader.subject = "Testing SMTP send ";

    const builder = new MessageBuilder();
    builder.header = testHeader;

    builder.htmlBody = "<div>Test message for send by SMTP.</div>";
    builder.textBody = "Test message for send by SMTP.\n-----\n";

    return builder.data;
};

const message = createMessage();

//Вызываем метод отправки со сформированным сообщением
session.sendMessageOperation(
    testConnection.EMAIL,
    [testConnection.MAIL_RECIPIENT],
    message
    ).then((result) => {
            try {
                    done();
            } catch (e) {
                    done(e);
            }
    }).catch((e) => {
    done(e);
});
```

2. Работа с протоколом IMAP (письма и папки)

trusted-mailcore2 поддерживает работу с протоколом IMAP, что позволяет получать письма с сервера, управлять папками, помечать письма прочитанными и удалять их. В примере вызовем некоторые из доступных методов.

```
const session = new IMAPSession();

session.hostname = "test.outlook.com";
session.port = 993;
session.username = "test@hotmail.com";
```

```
session.password = "test";
session.connectionType = ConnectionType.ConnectionTypeStartTLS;
session.checkCertificateEnabled = true;

//Получим все подписанные папки
const folders = session.fetchSubscribedFolders();

//Получим и выведем статус папки INBOX
const folderStatus = session.folderStatus("INBOX");
console.log(folderStatus.unseenCount);
console.log(folderStatus.messageCount);
console.log(folderStatus.recentCount);
console.log(folderStatus.uidNext);
console.log(folderStatus.uidValidity);
console.log(folderStatus.highestModSeqValue);

//Получим сообщения
const messages = sessionLogined.fetchMessagesByUID("INBOX");

//Скопируем сообщение из одной в папки в другую по его UID
const result = sessionLogined.copyMessages("INBOX", "PERSONAL", 1, 100);
```

Преимущества использования trusted-mailcore2

Высокий уровень безопасности:

1. Поддержка SSL/TLS обеспечивает защиту данных при взаимодействии с почтовыми серверами.
2. Поддержка российских криптографических стандартов: важным преимуществом является поддержка ГОСТ для S/MIME, что позволяет подтвердить авторство письма и сохранить конфиденциальность переписки.

Работа в Node.js:

Модуль создан специально для экосистемы Node.js, что позволяет легко интегрировать его в существующую архитектуру без написания сложного кода.

Как начать работать с trusted-mailcore2

Чтобы начать использование **trusted-mailcore2** в вашем проекте на Node.js, достаточно установить модуль через npm или скачать с сервера и следовать инструкциям в документации. Если вам нужна помощь с настройкой или интеграцией, наша команда готова оказать поддержку, обращайтесь по адресу sales@cryptoarm.ru.

Заключение

КриптоАРМ Сервер и trusted-mailcore2 — это безопасное решение для работы с электронной почтой на платформе Node.js. Благодаря расширенной безопасности, поддержке почтовых протоколов и лёгкости интеграции этот модуль становится незаменимым инструментом для серверных приложений, работающих с электронной почтой.

Попробуйте **trusted-mailcore2** и убедитесь в его преимуществах сами. Мы уверены, что этот инструмент упростит разработку и повысит надёжность вашего проекта.

Если у вас остались вопросы или нужна помощь в автоматизации работы с почтой, свяжитесь с нами по электронной почте sales@cryptoarm.ru.

Открыть статью на сайте

Сервис проверки и улучшения электронной подписи

В статье рассматриваются различные форматы фодписи PadES, CadES и XadES. Более подробно освещается стандарт CAdES (CMS Advanced Electronic Signatures) для создания улучшенной подписи. Предлагается вариант создания сервиса проверки и улучшения подписи на основе программного продукта КриптоАРМ Server. В качестве примера иллюстрируется проверка подписи и ее улучшение через пользовательский интерфейс сервиса.

Форматы подписи PadES, CadES и XadES

`PadES`, `CadES` и `XadES` — это наиболее распространенных форматы электронной подписи:

1. **Формат PadES**

   `PAdES` означает `PDF Advanced Electronic Signature` и является самым простым форматом. Фактически, присоединение электронной подписи `PAdES` к PDF создает криптографический конверт, который сам является PDF документом. Другими словами, электронная подпись `PAdES` — это PDF.

   У него есть несколько достоинств:

   - нет программного обеспечения или ридера для открытия PDF-конверта,
   - позволяет подписывать документ, не делая недействительными предыдущие подписи,
   - позволяет добавлять в документ видимую графическую подпись в дополнение к цифровой подписи.

   Однако `PAdES` позволяет подписывать только файлы PDF.

2. **Формат CadES**

   `CAdES` означает `CMS Advanced Electronic Signature (расширенная электронная подпись CMS)`, а `CMS` означает «криптографический синтаксис сообщения».

   После прикрепления подписи `CAdES` генерируется криптографический конверт, содержащий исходный файл. Расширение этого нового файла — `.p7m`.

   Эту электронную подпись можно прикрепить к любому типу файла : текстовым файлам (.doc, .docx), электронным таблицам (.xls, .xlsx), изображениям (.jpg, .png, .gif) и PDF-файлам. Эта характеристика делает ее универсальной и широко используемой подписью.

   Однако у `CAdES` есть недостатки:

   - пользователям необходимо специальное программное обеспечение для цифровой подписи , чтобы открыть конверт `p7m`,

- в случае наличия нескольких подписей на одном документе пользователям приходится заново создавать конверт p7m для каждой проставленной подписи (создавая так называемый эффект матрешки),
- не позволяет добавлять видимую графическую подпись на документ.

3. **Формат XAdES**

   XAdES означает XML Advanced Electronic Signature (расширенная электронная подпись XML), а XML — расширяемый язык разметки. Фактически, это стандарт для электронного подписания документов в формате XML.

   XAdES считывается как человеком, так и компьютером и используется для подписи электронных документов, таких как изображения (. jpeg, .png), звуковые файлы .mp3 , двоичные данные (.exe), PDF-файлы.

   С помощью XAdES можно выполнить подпись двумя способами:

   - Создав XML- файл для подписи без изменения исходного файла, таким образом, чтобы подпись и подписываемые данные были разделены;
   - Создав XML- файл, включающий подпись и данные.

   Этот формат цифровой подписи имеет свои преимущества:

   - не требует процесса конвертирования,
   - позволяет ставить несколько подписей, не делая предыдущие недействительными и не создавая эффекта матрешки.

Стандарт и формат подписи CAdES

CAdES определяет восемь профилей, которые предлагают различные уровни защиты для использования с подписанными данными CMS. С CAdES документы с электронной подписью могут оставаться действительными в течение многих лет, даже если подписывающий или проверяющий документ попытается оспорить его действительность в будущем.

В соответствии со спецификациями ETSI TS 101 733 V2.2.1 (2013-04) подписывающая сторона должна создать свою подпись в одном из двух форматов: базовая электронная подпись CAdES (CAdES-BES) или электронная подпись с явной политикой CAdES (CAdES -EPES).

Подпись, созданная с помощью базовой электронной подписи CAdES, будет содержать:

- Подписанные данные пользователя (отправляемый документ).
- Коллекция обязательных подписанных атрибутов:
  - Тип контента;
  - Дайджест сообщения;
  - Сертификат подписи ESS или сертификат подписи ESS -v2.
- Дополнительные обязательные подписанные атрибуты (как определено в отправленном документе).

- Значение цифровой подписи, вычисленное на основе пользовательских данных и подписанных атрибутов.

Кроме того, базовая электронная подпись CAdES может содержать необязательные подписанные атрибуты, включая:

- Время подписания (Signing-time);
- Комментарии (Content-hints);
- Ссылки (Content-reference);
- Идентификатор (Content-identifier);
- Указание типа обязательства (Commitment-type-indication);
- Местоположение подписавшего (Signer-location);
- Атрибуты подписчика (Signer-attributes);
- Штамп времени (Content-time-stamp);
- Тип кодировки MIME (Mime-type).

Электронная подпись CAdES Explicit Policy использует подписанный атрибут (sigPolicyID) для дальнейшего расширения определения электронной подписи с целью ее соответствия идентифицированной политике подписи.

Политика подписи

Политики подписи, связанные с CAdES, могут использоваться для установления согласованности проверенных электронных подписей. Когда проверяющий использует комплексную политику подписи, указанную подписывающим или подразумеваемую подписанными данными, он получит согласованный результат при попытке проверки электронной подписи.

Однако, если подписывающий или подписанные данные не указывают, какая политика подписи была использована, или если политика подписи выглядит неполной, результаты, достигнутые проверяющими, могут быть непоследовательными. Рекомендуется, чтобы для поддержания согласованности и подписывающий, и проверяющий согласовали одну и ту же всеобъемлющую политику подписи.

Дополнительные данные проверки

Дополнительные данные проверки могут быть добавлены в форматы CAdES-BES и CAdES-EPES для дальнейшей проверки электронной подписи. Эти данные собираются подписывающим лицом или проверяющим лицом, или обоими. Эти дополнительные данные включают:

- Сертификаты открытых ключей (PKC);
- Информация о статусе отзыва для отдельных PKC;
- Доверенная временная метка в дополнение к временной метке по умолчанию;
- Если применимо, информация о политике подписи, используемой для проверки электронной подписи.

В зависимости от выбранных выше проверочных данных при использовании форматов CAdES-BES или CAdES-EPES для создания электронной подписи применяются форматы CAdES:

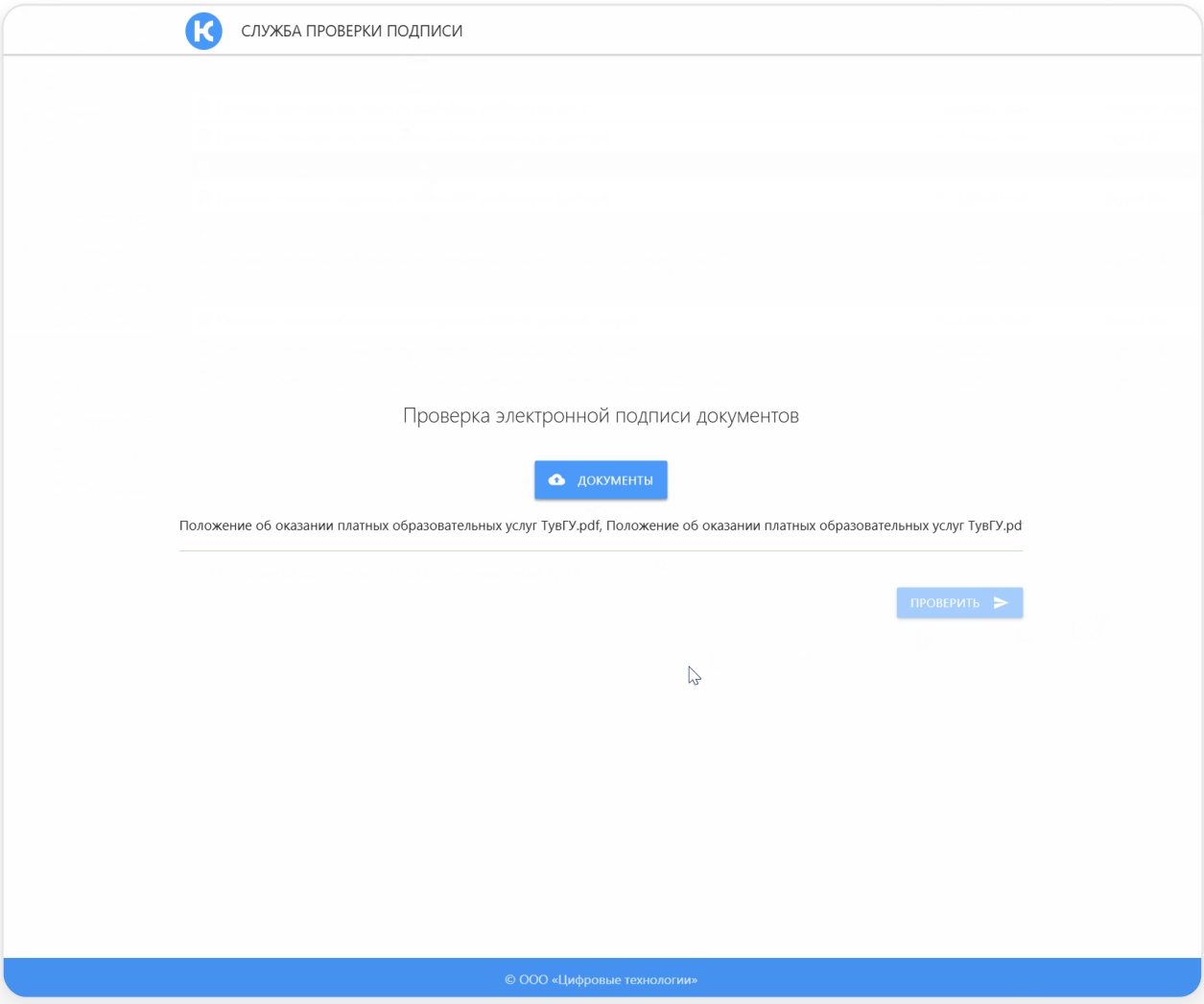
- CAdES-T – электронная подпись со временем. Этот формат применяется, когда доверенное время связано с электронной подписью либо путем добавления неподписанного атрибута временной метки, либо поставщик доверенных услуг предоставляет временную метку.
- CAdES-C – электронная подпись с полными ссылками на данные проверки. Этот формат добавляет атрибуты complete-certificate-references и complete-revocation-references к CAdES-T.
- CAdES-C – расширенные форматы электронной подписи. Форматы этой категории расширяют проверку CAdES-C, добавляя неподписанные атрибуты к электронной подписи, которые используются, когда требуется проверка подписанного документа в течение очень длительного периода.
    - CAdES-X Long – расширенная длинная электронная подпись. Эта опция добавляет атрибуты certificate-values и revocation-values, которые предоставляют информацию о сертификате и отзыве для проверки и средства предотвращения потери этой информации.
    - CAdES-X Type 1 – расширенная электронная подпись с типом времени 1. Эта опция добавляет атрибут временной метки, основанный на токене временной метки, созданном с помощью CAdES-C, и может защитить ключи от компрометации.
    - CAdES-X Type 2 — расширенная электронная подпись с типом времени 2. Этот параметр добавляет атрибут CAdES-C-time-stamped-cert-crls-reference, который предоставляет токен временной метки в отношении пути сертификации и информации об отзыве для защиты ключей от будущей компрометации.
    - CAdES-X Long Type 1 или 2 – расширенная электронная подпись со временем. Этот вариант объединяет CAdES-X Long либо с CAdES-X Type 1, либо с CAdES-X Type 2.
- CAdES-A – архивная электронная подпись. Форматы этой категории обеспечивают обратную совместимость и сохранение долгосрочной проверки подписи для архивных документов.
    - CAdES-A – архивная электронная подпись с атрибутом archive-time-stamp (ATSv2). Эта опция добавляет один или несколько атрибутов archive-time-stamp к CAdES-X Long или CAdES-X Long Type 1 или 2.
    - CAdES-A – архивная электронная подпись с атрибутом archive-time-stamp (ATSv2). Эту опцию можно добавить к CAdES-BES, CAdES-EPES, CAdES-T, CAdES-C, CAdES-X, CAdES-XL или CAdES-A для долгосрочного архивирования подписей.
- CAdES-LT – Долгосрочная электронная подпись. Этот формат добавляет один или несколько атрибутов долгосрочной проверки к CAdES-T, CAdES-C, CAdES-X Long, AdES-X Long Type 1 или 2 или CAdES-A, чтобы обеспечить архивирование долгосрочных подписей. Его можно использовать для защиты всего документа и подписи перед отправкой в службу сохранения.

Полное техническое описание расширенных электронных подписей CMS см. в документе ETSI TS 101 733 Электронная подпись и инфраструктура (ESI) – Расширенная электронная подпись CMS (CAdES).

Сервис проверки и улучшения подписи на SDK КриптоАРМ Server

С помощью программного продукта `КриптоАРМ Server` (используется SDK продукта) можно реализовать сервис проверки электронной подписи для различных видов документов и предоставить функцию создания улучшеной подписи, используя форматы `CAdES`.

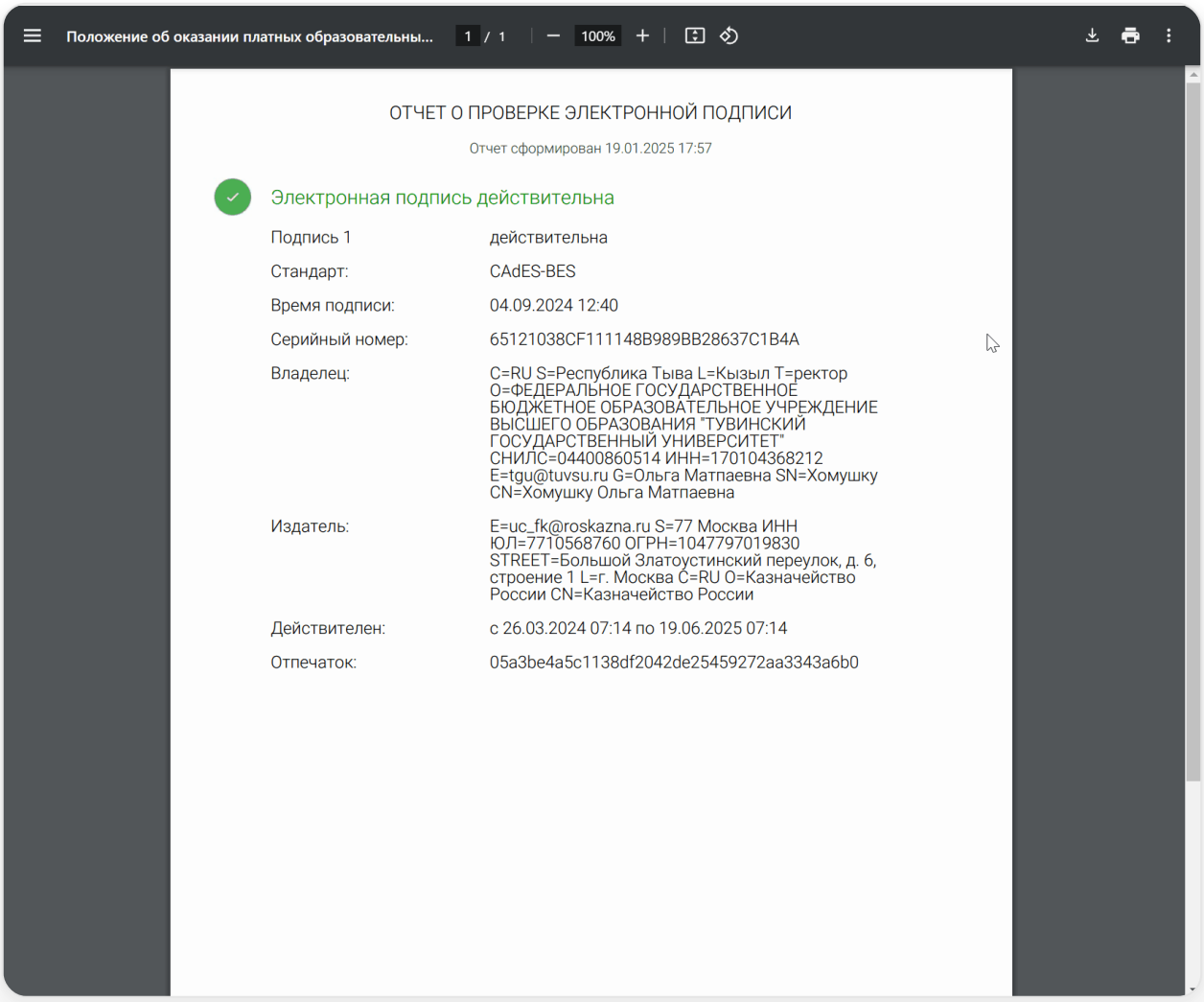
Рассмотрим один из вариантов программного интерфейса подобного сервиса. Ниже представлен интерфейс загрузки документов для проверки и улучшения подписи. Используя этот интерфейс пользователь может выбрать и загрузить набор документов (подпись документов может быть как отделенная, так и присоединенная).



Статичный кадр демонстрации. Открыть анимацию

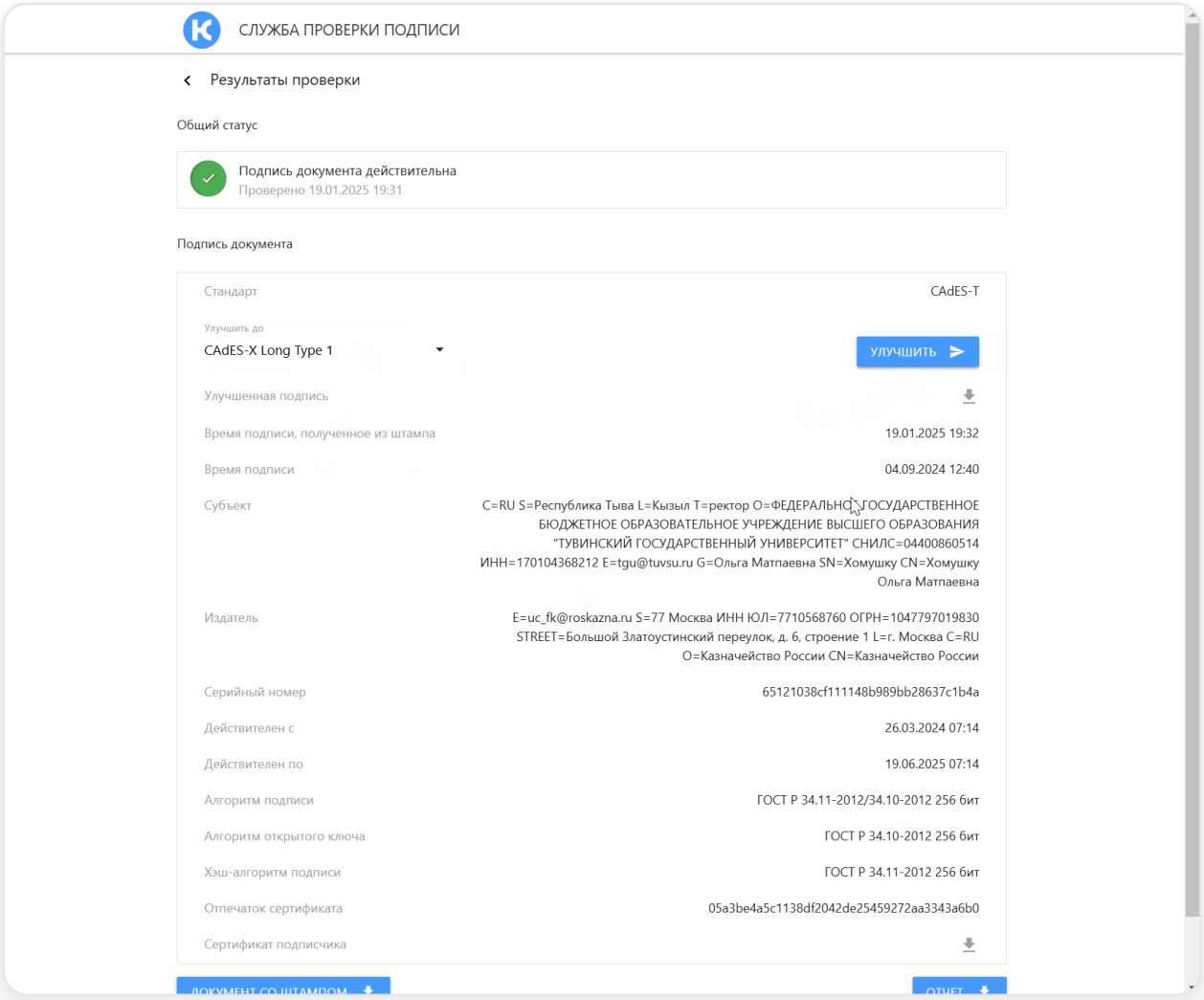
В примере использованы два файла - один исходный документ формата PDF, другой - отделенная подпись этого документа. После выполнения команды проверки подписи формируется отчет, где приводятся результаты валидности подписи (с отображением атрибутов), а также определяется ее формат. Со страница результатов проверки подписи можно скачать сертификат подписчика, документ с наложенным визуальным штампом времени (визуальное представление штампа на первой странице свидетельствует о наличии электронной подписи документа), посмотреть доступные форматы CAdES и выполнить улучшение до требуемого формата и скачать отчет о подписи документа в формате PDF.

На следующей иллюстрации показано скачивание отчета о подписи в формате PDF-документа.



Статичный кадр демонстрации. Открыть анимацию

Для улучшения подписи пользователь может выбрать требуемый формат подписи CAdES из предложенного списка на странице результатов проверки подписи и выьрать команду ее улучшения. Подпись документа будет преобразована к этому формату. В примере показоно улучшение подписи с формата CAdES-BES до формата CAdES-T.



Статичный кадр демонстрации. Открыть анимацию

Сервис проверки и улучшения электронной подписи формата CAdES, в сочетании с использованием штампов времени, представляет собой мощный инструмент для обеспечения безопасности и доверия в цифровом документообороте. В условиях стремительного роста объемов электронных транзакций и документации, надежная электронная подпись становится критически важной для защиты интересов как бизнеса, так и частных лиц.

Интеграция штампов времени в процесс подписания документов добавляет дополнительный уровень уверенности, позволяя точно фиксировать момент подписания и подтверждая юридическую силу документа на протяжении всего его срока действия. Это особенно актуально в контексте соблюдения нормативных требований и защиты прав сторон.

Таким образом, использование сервиса проверки и улучшения электронной подписи формата CAdES вместе со штампами времени не только повышает уровень безопасности и доверия к цифровым документам, но и способствует более эффективному и прозрачному ведению дел.

Первоисточники

1. Избранные статьи о цифровых подписях (2014–2016 гг.) Ашика Д, Дона М. Тернера, Гийома Форже, Питера Лэндрока и Торбена Педерсена
2. Trust Services и eID (получено 11.01.2016) Европейской комиссией
3. РЕГЛАМЕНТ (ЕС) № 910/2014 об электронной идентификации и трастовых услугах для электронных транзакций на внутреннем рынке и об отмене Директивы 1999/93/EC (2014) Европейского парламента и Европейской комиссии
4. Электронные подписи и инфраструктуры (ESI) - CMS Advanced Electronic Signature (CAdES). (2013) Европейского института стандартов в области телекоммуникаций

Открыть статью на сайте

Часто задаваемые вопросы

Для чего нужен КриптоАРМ Server

КриптоАРМ Server — программный продукт, предназначенный для интеграции и автоматизации криптографических операций в информационных системах организаций. Он включает в себя набор криптографических библиотек и средства разработки (SDK). Подробнее читайте в статьте О продукте.

Какие криптографические операции поддерживает КриптоАРМ Server

- Формирование электронной подписи, проверка подписи, шифрование и расшифрование, сведения о сертификатах, генерация ключевой пары и создание запросов на получение сертификата, подключение защищенных носителей.
- Подпись и проверка электронной подписи встроенной в pdf, разметка областей для оттисков подписи, добавление графических штампов с информацией о подписи, пакетная подпись, сертификация подписанных документов с защитой от изменений, конвертация pdf-файлов в формат pdf-a/2b.
- Сбор, разбор и отправка электронных писем, поддержка IMAP, SMTP, POP3, S/MIME, контакты и адресные книги vCard, CardDAV.
- Обеспечение защищенного соединения к веб-серверам с поддержкой ГОСТ TLS.
- Проверка электронной подписи формата XMLDSig (XML Digital Signature).

Как интегрировать КриптоАРМ в свою систему

КриптоАРМ предлагает два инструмента для интеграции и автоматизации: API и SDK. Оба варианта упрощают автоматизацию криптографических операций, но имеют разные области применения. Читайте подробнее в нашей статье: Что такое КриптоАРМ API и SDK.

Где посмотреть документацию и примеры для API или SDK?

1. Документация КриптоАРМ API и КриптоАРМ SDK.
2. Пример по вызову API реализован на этом стенде, его исходный код - ссылка.
3. В качестве примера использования КриптоАРМ SDK мы реализовали сервис по проверке подписи и постановке визуальных штампов. Сервис доступен по ссылке, его исходный код открыт - ссылка.
4. Ссылка на видеодемонстрацию работы API от наших партнеров.

Мы готовы помочь вам с интеграцией API и SDK в вашу систему, по всем вопросам обращайтесь на sales@cryptoarm.ru.

Требуется ли оценка влияния?

Для использования КриптоАРМ в некоторых сценариях может потребоваться проведение исследований по оценке влияния (СФ).

В КриптоАРМ возможны следующие варианты применения:

1. Использование графических интерфейсов СКЗИ. В данном варианте использования СКЗИ дополнительных исследований **не требуется**.
2. Программный вызов КриптоАРМ API. В данном варианте использования СКЗИ **требуется проведение исследований** по оценке влияния СФ на СКЗИ.
3. Использование КриптоАРМ SDK рассматривается как встраивание СКЗИ во вновь разрабатываемое или существующее прикладное программное обеспечение, программно-аппаратные и аппаратные решения. В данном случае **требуется проведение тематических исследований** СФ со встроенным СКЗИ как самостоятельного шифровального (криптографического) средства.

Если вам требуется помощь в оценке влияния, оставьте заявку на sales@cryptoarm.ru.

Какие лицензии нужны

Для использования КриптоАРМ SDK необходима серверная лицензия.

КриптоАРМ API может работать как с клиентскими, так и серверными лицензиями на КриптоАРМ. С актуальными ценами можно ознакомиться в интернет-магазине cryptoarm.ru и в прайс-листе.

Для API будет доступен дополнительный вариант лицензирования — транзакционный. В этом случае оператор информационной системы приобретает пакет лицензий и распространяет своим клиентам клиентские лицензии вместе с параметрами операции. Пользователю не надо покупать КриптоАРМ или вручную вводить лицензии.

Также потребуется лицензия на криптопровайдер КриптоПро CSP.

Как интегрировать проверку подписи в свою систему

С помощью КриптоАРМ Server можно легко интегрировать проверку электронной подписи во внешние системы. КриптоАРМ предлагает два инструмента для интеграции и автоматизации: API и SDK. Оба варианта упрощают автоматизацию криптографических операций, но имеют разные области применения.

Читайте подробнее в наших статьях:

- Что такое КриптоАРМ API и SDK,

- Автоматизация проверки электронной подписи с использованием КриптоАРМ Сервер.

Можно ли создать протокол проверки подписи

С помощью КриптоАРМ Server можно создать протокол (отчет) проверки электронной подписи. Кроме того, для PDF-файлов доступно создание копии с визуальным штампом.

Примеры реализации описаны в наших статьях:

- Сервис проверки и визуализации электронной подписи,
- Сервис проверки и улучшения электронной подписи.

Можно ли создать копию документа с визуальным штампом

С помощью КриптоАРМ Server можно проверить электронную подпись и визуализировать штамп на PDF-документе. Пример реализации описан в нашей статье Сервис проверки и визуализации электронной подписи.

Как интегрировать электронную подпись в свою систему

С помощью КриптоАРМ Server можно автоматизировать процессы подписания электронных документов. Для этого КриптоАРМ предлагает два инструмента: API и SDK. Оба варианта упрощают автоматизацию криптографических операций, но имеют разные области применения.

Читайте подробнее в наших статьях:

- Что такое КриптоАРМ API и SDK,
- Автоматическое подписание обезличенным сертификатом,
- Интеграция КриптоАРМ во внешнюю информационную систему.

Какие стандарты подписи поддерживаются

В КриптоАРМ Server реализована поддержка стандартов CMS, CAdES, PAdES.

Можно ли улучшить электронную подпись с помощью КриптоАРМ Server

С помощью КриптоАРМ Server можно улучшить любую электронную подпись с формата CAdES-BES до CAdES-A.

- Пример реализации описан в нашей статье Сервис проверки и улучшения электронной подписи.
- Ссылка на видеообзор от наших партнеров.

Как защитить документы с помощью КриптоАРМ Server

С помощью КриптоАРМ Server можно автоматизировать процессы шифрования и расшифрования файлов.

Кроме того, с помощью КриптоАРМ Server (модуль trusted-mailcore2) можно реализовать интеграцию электронной почты в вашем проекте на Node.js. Основные преимущества такой интеграции:

- Криптографическая защита писем: поддержка стандарта S/MIME с использованием ГОСТ-алгоритмов,
- Надежные TLS соединения: благодаря интеграции с cpcurl от КриптоПро обеспечена поддержка ГОСТ TLS для безопасных соединений с почтовыми серверами.
- Работа с сертифицированным криптопровайдером КриптоПро CSP.

Читайте подробнее:

- Пример использования описан в статье Работа с почтой в Node.js. Примеры и возможности КриптоАРМ Сервер.
- Документация КриптоАРМ API и КриптоАРМ SDK.

Открыть статью на сайте

Глоссарий

Сертификаты

Термин	Определение
Сертификат электронной подписи	Электронный документ, который подтверждает владение ключом электронной подписи
Сертификат усиленной квалифицированной электронной подписи (УКЭП)	Выпускает только аккредитованный удостоверяющий центр. Документ, подписанный УКЭП, действителен для любой организации или ведомства.
Самоподписанный сертификат	Сертификат, изданный самим пользователем, без обращения к Удостоверяющему центру.
Открытый ключ электронной подписи	Ключ проверки электронной подписи. Содержит содержит уникальный код, с помощью которого можно проверить подлинность электронной подписи. Сертификат открытого ключа содержит сведения о подписанте (ФИО, СНИЛС и др.), об удостоверяющем центре, который выдал ЭП и срок действия ключа.
Закрытый ключ электронной подписи	Секретный уникальный набор символов, который также формируется криптопровайдером. Закрытый ключ необходим для формирования электронной подписи и хранится в зашифрованном виде, как правило, на носителе (токене). Доступ к закрытому ключу имеет только владелец.
Запрос на сертификат	Электронный документ, который содержит информацию для получения сертификата электронной подписи в Удостоверяющем центре
Ключевой контейнер	Способ хранения закрытых ключей. Кроме ключей, он содержит служебную информацию, необходимую для обеспечения криптографической защиты ключей, их целостности и т.п. Физическое представление ключевого контейнера зависит от типа ключевого носителя: флешка, жёсткий диск, смарт-карты, реестр.
Статус сертификата электронной подписи	Информация о действительности сертификата
Удостоверяющий центр (УЦ)	Доверенная организация, которая имеет право выпускать сертификаты электронной подписи (ЭП) юридическим и физическим лицам

Термин	Определение
Корневой сертификат	Цифровой документ, в котором содержится информация об удостоверяющем центре. Корневые сертификаты выдаются только организациям и могут иметь разный уровень. В Российской Федерации главный корневой сертификат принадлежит Министерству цифрового развития. На основе этого сертификата Минцифры РФ выдаёт удостоверяющим центрам собственные сертификаты, подписывая их своим корневым сертификатом.
Промежуточный сертификат	Сертификат, выданный удостоверяющим центрам
Сертификат другого пользователя	Сертификат, принадлежащий другому пользователю. Предназначен для шифрования документов в его адрес.
Список отзыва сертификатов	Список сертификатов, которые удостоверяющий центр пометил как отозванные.
Цепочка сертификации или цепочка доверия	Иерархическая коллекция сертификатов. Для квалифированных сертификатов ЭП, как правило, состоит из 3 звеньев. Верхнее звено - корневой сертификат. Среднее звено - сертификат удостоверяющего центра. Конечное звено - сертификат пользователя.
Сервис DSS	Сервис электронной подписи. Предназначен для шифрования и расшифрования электронных документов, формирования запросов на создание и управление сертификатами ключей проверки электронной подписи, создания и проверки электронной подписи различного формата криптографических сообщений.

Подпись и шифрование

Термин	Определение
Электронная подпись (ЭП)	Информация в электронно-цифровой форме для определения и подтверждения личности автора подписи
Усиленная неквалифицированная электронная подпись (УНЭП)	Цифровой аналог собственноручной подписи. Применяется, если стороны заключили соглашение об этом или так указано в законе. В таких случаях документы, подписанные от руки или с помощью УНЭП, имеют равную юридическую силу.

Термин	Определение
Усиленная квалифицированная электронная подпись (УКЭП)	Цифровой аналог собственноручной подписи. Ей можно подписывать любые документы без предварительной договорённости с другой стороной.
Профили подписи в КриптоАРМ	Заранее подготовленные наборы настроек, которые можно применять для выполнения подписи, архивирования и шифрования.
Алгоритмы подписи	Криптографические алгоритмы, которые используются для создания электронной цифровой подписи
Алгоритмы шифрования	Криптографические алгоритмы, которые используются для шифрования файлов
Криптопровайдер	Провайдер криптографических услуг, криптопровайдер специализированный программный модуль или набор модулей, предоставляющих средства шифрования и дешифрования данных, цифровой подписи, а также генерации и хранения ключей.
Присоединенная подпись	Подпись, в процессе создания которой формируется специальный контейнер, в котором находятся как подписываемый документ, так и файл подписи.
Отсоединенная подпись	Файл подписи формируется и хранится отдельно от подписываемого документа.
Стандарт подписи CAdES-BES (Basic Electronic Signature)	Основной и простейший формат электронной подписи, описываемый в стандарте CAdES. Он обеспечивает базовую проверку подлинности данных и защиту их целостности
Стандарт подписи CAdES-T (Timestamp)	Формат электронной подписи, в которую автоматически добавляется штамп времени на подпись.
Стандарт подписи CAdES-X Long Type 1	Формат электронной подписи, включающий дополнительные атрибуты certificate-values и revocation-values, которые представляют собой полные данные сертификатов и списки отзыва сертификатов. А также штампы времени на подпись и ссылки на сертификаты и списки отзыва сертификатов.
Стандарт подписи CAdES-A (Archival)	Формат электронной подписи, сформированный на основе CAdES-X Long Type 1 или Type 2 путём добавления одного или нескольких атрибутов archive-time-stamp, представляющих собой архивные штампы времени.

Термин	Определение
Машиночитаемая доверенность	Аналог бумажной доверенности в электронном виде. Она создаётся в формате XML, который позволяет компьютеру прочитать её содержание и подтвердить личность того, кто подписал документ
Сертификация PDF документа	Позволяет защитить PDF документ от изменений путем его подписания электронной подписью, с указанием типа изменений, разрешенных для документа.
Конвертация PDF документа	При конвертации PDF документа происходит удаление скриптов, форм и прочих элементов, влияющих на безопасность и целостность документа.
Формат подписи CadES	Стандарт электронной подписи, представляющий собой расширенную версию стандарта электронной подписи CMS. Эту электронную подпись можно прикрепить к любому типу файла : текстовым файлам (.doc, .docx), электронным таблицам (.xls, .xlsx), изображениям (.jpg, .png, .gif) и PDF-файлам.
Формат подписи PadES	Формат, при котором подпись встраивается в PDF-документ. «Встроенная» подпись позволяет подписывать электронные документы без изменения формата файла и без создания отдельного файла подписи, как это происходит при использовании присоединенной и отсоединенной подписи соответственно. Подпись встраивается в электронный документ, и получатель может проверить документ на действительность электронной подписи.
Видимая подпись PDF документа	«Встроенная» подпись, которая отображается на странице документа в виде штампа.
Невидимая подпись PDF документа	«Встроенная» подпись, которая визуально на страницах не видна.
Область для штампа подписи в PDF документе	Область на странице PDF документа, размеченная для видимой подписи.
Произвольная область в PDF документе	Область для графического изображения или текста. В области можно разместить свой комментарий, статус или отметку о конфиденциальности документа.

Почта

Термин	Определение
Адрес электронной почты	Название электронного почтового ящика, который может получать (и отправлять) электронные письма в сети (например, в интернете или в локальной сети, не подключённой к интернету).
Почтовый сервер	Это программа, работающая у интернет-провайдеров и на крупных сайтах и используемая для передачи почты. Обычно пользователи не взаимодействуют с почтовыми серверами напрямую: электронная почта отправляется с помощью почтового клиента на почтовый сервер, который доставляет её в почтовый клиент получателя.
Почтовый клиент	Это программа (например, на компьютере или мобильном устройстве), которая используется для чтения и отправки электронных сообщений.
IMAP	Протокол, который используется для получения сообщений электронной почты с сервера. IMAP позволяет почтовым программам получать доступ не только к новым сообщениям, но и к папкам на сервере. Действия синхронизируются между несколькими почтовыми программами, подключёнными через IMAP.
POP	Протокол, который используется для получения сообщений электронной почты с сервера. В отличие от IMAP, POP позволяет почтовому клиенту загружать только последние сообщения, которыми можно управлять в клиенте и на устройстве.
SMTP	Протокол, который используется для передачи электронной почты. Он определяет формат сообщений и процедуру их передачи через Интернет от отправителя к получателю с помощью почтовых серверов.
S/MIME	Стандарт для шифрования и подписи электронных писем с помощью открытого ключа.
SSL	Протокол, который обеспечивает безопасное соединение между клиентом и сервером.
Логин (имя пользователя)	Имя почтового ящика — комбинация из букв и цифр, которая, как правило, находится в адресе перед именем домена.
Пароль	Набор символов, предназначенный для подтверждения личности.

Термин	Определение
Скрытая копия	Копия электронного письма, отправленного получателю, адрес электронной почты которого не указан (в качестве получателя) в сообщении
Копия	Копия электронного письма, отправленного получателю, адрес электронной почты которого указан в поле заголовка «Копия».
Тело письма	Основная часть электронного письма, которая содержит текст, изображения и другие данные (например, прикреплённые файлы).
LDAP	Протокол быстрого доступа к каталогам. Он позволяет получать и принимать через сеть данные из каталогов с иерархической структурой.
ALD Pro	Служба каталога для Linux, программный комплекс централизованного управления компьютерами с помощью групповых политик, а также прикладными сервисами для ИТ-инфраструктур организаций различного масштаба. Комплекс создан «Группой Астра».
CardDAV	Протокол адресной книги, предназначенный для предоставления пользователям доступа к данным на сервере.
vCard	текстовый формат для обмена электронными визитными карточками. В карточке может содержаться имя, номера телефонов, адрес электронной почты, адрес проживания, дата рождения, фотография, сертификат и другие данные контакта.
Gravatar	Сервис компании Automattic, позволяющий интернет-пользователям хранить свой аватар на специальном сервере.

Открыть статью на сайте

Описание API модуля trusted-crypto

В данном документе описывается typescript интерфейс модуля trused-crypto. Для использования этого SDK в составе продукта КриптоАРМ.

В модуле представлены пространства имен (namespace):

1. **CMS** – набор классов для выполнения операций с цифровой подписью.
2. **PKI** – набор классов для работы с криптографическими объектами, такими как цифровые сертификаты, списки отзыва, идентификаторы, ключевые контейнеры и т.п.
3. **PKISTORE** – набор классов для работы с криптографическими провайдерами, посредством предоставляемого API.
4. **UTILS** – набор классов для работы с лицензией и логированием операций.
5. **COMMON** – набор методов для управления контекстом библиотеки OpenSSL.

Открыть статью на сайте

Описание класса SignedData

Метод	Возвращаемый тип	Описание
`load`	void или SignedData для static	Чтение подписанного файла
`sign` или `signAsync`	void	Создает подпись
`import` или `importAsync`	Void или SignedData для static	Импорт подписи из памяти
`export` или `exportAsync`	Buffer	Возвращает подписанное сообщение
`save` или `saveAsync`	void	Запись в файл
`verify` или `verifyAsync`	boolean	Проверка подписанного сообщения
get `content`	ISignedDataContent	Возвращает контент подписанного сообщения
set `content`	undefined	Задает контент подписанного сообщения
get `policies`	string[]	Возвращает политики
set `policies`	undefined	Задает политики
`freeContent`	void	Освобождение подписанного контента
`isDetached`	boolean	Проверка открепленная ли подпись
`certificates`	CertificateCollection или Certificate	Возвращает коллекцию сертификатов или сертификат подписчика
`signers`	SignerCollection или Signer	Возвращает подписчиков или подписчика
get `signParams`	TimestampParams или CadesParams	Возвращает параметры подписи
set `signParams`	void	Задает параметры подписи
`writeContent`	void	Запись контента в файл (асинхронно)
`isContentAvailable`	boolean	Проверка доступности контента

Открыть статью на сайте

Метод load

Чтение подписанного файла.

```
load(filename: string, format?: DataFormat): void
```

Асинхронный метод:

```
loadAsync(filename: string, format: DataFormat, done: (message: string) => void): void
```

Параметр	Тип	Описание
filename	string	Полный путь до файла
format?	DataFormat	Необязательный параметр. Тип кодировки.

Статичный метод `load` возвращает SignedData.

Открыть статью на сайте

Метод sign

Формирование подписанного сообщения.

```
sign(cert: pki.Certificate): void
```

Асинхронный метод:

```
signAsync(cert: pki.Certificate, done: (message: string) => void): void
```

Параметр	Тип	Описание
cert	pki.Certificate	Сертификат подписчика

Открыть статью на сайте

Метод import

Импорт подписанного сообщения из памяти.

```
import(buffer: Buffer, format: DataFormat = DEFAULT_DATA_FORMAT): void
```

Асинхронный метод:

```
importAsync(buffer: Buffer, format: DataFormat, done: (message: string) => void): void
```

Параметр	Тип	Описание
buffer	Buffer	Буфер с подписанным сообщением
format?	DataFormat	Необязательный параметр. Тип кодировки. Значение по умолчанию: DataFormat.DER

Статичный метод `import` возвращает SignedData.

Открыть статью на сайте

Метод export

Экспорт подписанного сообщения в буфер.

```
export(format: DataFormat = DEFAULT_DATA_FORMAT): Buffer
```

Асинхронный метод:

```
exportAsync(format: DataFormat = DEFAULT_DATA_FORMAT, done: (error: string, result: Buffer) =>
void): void
```

Параметр	Тип	Описание
format	DataFormat	Необязательный параметр. Тип кодировки. Значение по умолчанию: DataFormat.DER

Открыть статью на сайте

Метод save

Запись подписанного сообщения в файл.

```
save(filename: string, format: DataFormat): void
```

Асинхронный метод:

```
saveAsync(filename: string, format: DataFormat, done: (message: string) => void): void
```

Параметр	Тип	Описание
filename	string	Полный путь до файла
format	DataFormat	Тип кодировки (BASE64 или DER)

Открыть статью на сайте

Метод verify

Проверка подписи.

```
verify(signer?: cms.Signer): boolean
```

Асинхронный метод:

```
verifyAsync(done: (error: string, result: boolean) => void, signer?: cms.Signer): void
```

Параметр	Тип	Описание
signer?	cms.Signer	Объект подписчика, подпись которого будет проверяться. Необязательный параметр.

Открыть статью на сайте

Метод content

Получение или установка контента в объект подписи (getters/setters).

```
get content(): ISignedDataContent
```

```
set content(data: ISignedDataContent)
```

Параметр	Тип	Описание
data	ISignedDataContent	Контент (исходное сообщение)

Открыть статью на сайте

Метод policies

Получение или установка политик в объект подписи (getters/setters).

```
get policies(): string[]
```

```
set policies(values: string[])
```

Параметр	Тип	Описание
values	string[]	Массив политик (SignedDataPolicy)

Открыть статью на сайте

Метод freeContent

Освобождение контента (памяти).

```
freeContent(): void
```

Открыть статью на сайте

Метод isDetached

Проверка, является ли подпись открепленной.

```
isDetached(): boolean
```

Открыть статью на сайте

Метод certificates

Получение всех сертификатов подписчиков или одного сертификата, если передан индекс.

```
certificates(index?: number): CertificateCollection | Certificate
```

Параметр	Тип	Описание
index?	number	Индекс сертификата, который надо получить. Необязательный параметр.

Открыть статью на сайте

Метод signers

Получение всех подписчиков или одного, если передан индекс.

```
signers(index?: number): SignerCollection | Signer
```

Параметр	Тип	Описание
index?	number	Индекс сертификата, который надо получить. Необязательный параметр.

Открыть статью на сайте

Метод signParams

Получение или установка дополнительных параметров подписи (getters/setters).

```
get signParams(): TimestampParams | CadesParams
```

```
set signParams(params: TimestampParams | CadesParams)
```

Параметр	Тип	Описание
params	TimestampParams или CadesParams	Параметры подписи, отвечающий за установку штампов времени и cades

Открыть статью на сайте

Описание класса Signer

Метод	Возвращаемый тип	Описание
get `certificate`	Certificate	Возвращает сертификат подписчика
set `certificate`	void	Задает сертификат подписчика
get `index`	number	Возвращает индекс подписчика
set `index`	void	Задает индекс подписчика
get `signingTime`	Date	Возвращает время подписи (из подписанных атрибутов)
get `signatureAlgorithm`	string	Возвращает алгоритм подписи
get `signatureDigestAlgorithm`	string	Возвращает хэш алгоритм подписи
get `issuerName`	string	Возвращает имя издателя сертификата подписчика
get `serialNumber`	string	Возвращает серийный номер сертификата подписчика
`timestamp`	TSP	Возвращает штамп времени по его типу
`verifyTimestamp`	boolean	Проверка штампа времени
`isCades`	boolean	Возвращает true если это CAdES
get `certificateValues`	CertificateCollection	Только для CAdES. Возвращает коллекцию сертификатов из атрибута certificateValues.
get `revocationValues`	Buffer[]	Только для CAdES. Возвращает массив с закодированными значениями отзыв (ответ OCSP или CRL).
get `ocspResp`	OCSP	Только для CAdES. Возвращает OCSP ответ.
`verifySignatureValue`	boolean	Проверка подписи по хешу

Открыть статью на сайте

Метод certificate

Возвращает сертификат подписчика.

```
get certificate(): Certificate
```

Открыть статью на сайте

Метод index

Получение или установка индекса подписчика (getters/setters)

```
get index(): number
```

```
set index(ind: number)
```

Параметр	Тип	Описание
ind	number	Индекс подписчика

Открыть статью на сайте

Метод signingTime

Возвращает время подписи (из подписанных атрибутов).

```
get signingTime(): Date
```

Открыть статью на сайте

Метод signatureAlgorithm

Возвращает алгоритм подписи.

```
get signatureAlgorithm(): string
```

Открыть статью на сайте

Метод signatureDigestAlgorithm

Возвращает хэш алгоритм подписи.

```
get signatureDigestAlgorithm(): string
```

Открыть статью на сайте

Метод issuerName

Возвращает имя издателя сертификата подписчика.

```
get issuerName(): string
```

Открыть статью на сайте

Метод issuerName

Возвращает серийный номер сертификата подписчика.

```
get serialNumber(): string
```

Открыть статью на сайте

Метод timestamp

Возвращает штамп времени по его типу.

```
timestamp(tspType: number): TSP
```

Параметр	Тип	Описание
tspType	number	Тип штампа времени

Открыть статью на сайте

Метод verifyTimestamp

Проверка штампа времени по типу.

```
verifyTimestamp(tspType: number): boolean
```

Параметр	Тип	Описание
tspType	number	Тип штампа времени

Открыть статью на сайте

Метод isCades

Проверка является ли подпись CAdES.

```
get isCades(): boolean
```

Открыть статью на сайте

Метод certificateValues

Только для CAdES. Возвращает коллекцию сертификатов из атрибута `certificateValues`.

```
get certificateValues(): CertificateCollection
```

Открыть статью на сайте

Метод revocationValues

Только для CAdES. Возвращает массив с закодированными значениями отзыва (ответ OCSP или CRL).

```
get revocationValues(): Buffer[]
```

Открыть статью на сайте

Метод ocspResp

Только для CAdES. Возвращает OCSP ответ.

```
get ocspResp(): OCSP
```

Открыть статью на сайте

Метод verifySignatureValue

Проверка подписи по хешу.

```
verifySignatureValue(hashValue: Buffer, signValue: Buffer, signAlg: string | undefined,
signCertOrKeyValue: pki.Certificate | Buffer, keyAlg?: string, namedCurve?: string): boolean
```

Параметр	Тип	Описание
hashValue	Buffer	Значение хеша
signValue	Buffer	Значение подписи
signAlg	string или undefined	OID алгоритма подписи
signCertOrKeyValue	pki.Certificate или Buffer	Сертификат подписчика или закодированная структура SubjectPublicKeyInfo
keyAlg	string	Алгоритм ключа. Необязательный параметр

Открыть статью на сайте

Описание класса SignerCollection

Метод	Возвращаемый тип	Описание
`items`	Signer	Возвращает объект коллекции по его индексу
get `length`	number	Возвращает количество элементов коллекции

Открыть статью на сайте

Метод items

Возвращает объект коллекции по его индексу.

```
items(index: number): Signer
```

Параметр	Тип	Описание
index	number	Индекс элемента в коллекции

Открыть статью на сайте

Метод length

Возвращает количество элементов коллекции.

```
get length(): number
```

Открыть статью на сайте

Описание класса CadesParams

Метод	Возвращаемый тип	Описание
get `cadesType`	number	Возвращает тип (стандарт) cades
set `cadesType`	void	Задает тип (стандарт) cades
get `connSettings`	ConnectionSettings	Возвращает настройки соединения для создания tsp
set `connSettings`	void	Задает настройки соединения для создания cades
get `tspHashAlg`	string	Возвращает хэш алгоритм
set `tspHashAlg`	void	Задает хэш алгоритм (OID)
get `ocspSettings`	ConnectionSettings	Возвращает настройки соединения для создания ocsp
set `ocspSettings`	void	Задает хэш алгоритм (OID)

Открыть статью на сайте

Метод cadesType

Получение или установка типа стандарта cades (getters/setters).

В текущей версии всегда 1 (CAdES-X Long Type 1).

```
get cadesType(): number
```

```
set cadesType(signType: number)
```

Параметр	Тип	Описание
signType	number	В текущей версии всегда 1 (CAdES-X Long Type 1)

Открыть статью на сайте

Метод connSettings

Получение или установка настроек соединения для создания cades (getters/setters).

```
get connSettings(): utils.ConnectionSettings
```

```
set connSettings(connSett: utils.ConnectionSettings): void
```

Параметр	Тип	Описание
connSet	utils.ConnectionSettings	Набор параметров, используемых для установки сетевого соединения (адрес, аутентификация, пароль и т.д.)

Открыть статью на сайте

Метод tspHashAlg

Получение или установка используемого алгоритма хеширования (getters/setters).

```
get tspHashAlg(): string
```

```
set tspHashAlg(hashAlgOid: string): void
```

Параметр	Тип	Описание
hashAlgOid	string	OID используемого хеш-алгоритма. Если не будет задан, то используется значение из сертификата

Открыть статью на сайте

Метод ocspSettings

Получение или установка настроек соединения для создания cades (getters/setters).

```
get ocspSettings (): utils.ConnectionSettings
```

```
set ocspSettings (connSett: utils.ConnectionSettings): void
```

Параметр	Тип	Описание
connSet	utils.ConnectionSettings	Набор параметров, используемых для установки сетевого соединения (адрес, аутентификация, пароль и т.д.)

Открыть статью на сайте

Описание класса TimestampParams

Метод	Возвращаемый тип	Описание
get `stampType`	number	Возвращает тип штампа времени
set `stampType`	void	Задает тип штампа времени
get `connSettings`	ConnectionSettings	Возвращает настройки соединения для создания tsp
set `connSettings`	void	Задает настройки соединения для создания cades
get `tspHashAlg`	string	Возвращает хэш алгоритм
set `tspHashAlg`	void	Задает хэш алгоритм (OID)

Открыть статью на сайте

Метод connSettings

Получение или установка настроек соединения для создания cades (getters/setters).

```
get connSettings(): utils.ConnectionSettings
```

```
set connSettings(connSett: utils.ConnectionSettings): void
```

Параметр	Тип	Описание
connSet	utils.ConnectionSettings	Набор параметров, используемых для установки сетевого соединения (адрес, аутентификация, пароль и т.д.)

Открыть статью на сайте

Метод tspHashAlg

Получение или установка используемого алгоритма хеширования (getters/setters).

```
get tspHashAlg(): string
```

```
set tspHashAlg(hashAlgOid: string): void
```

Параметр	Тип	Описание
hashAlgOid	string	OID используемого хеш-алгоритма. Если не будет задан, то используется значение из сертификата

Открыть статью на сайте

Метод ocspSettings

Получение или установка настроек соединения для создания cades (getters/setters).

```
get ocspSettings (): utils.ConnectionSettings
```

```
set ocspSettings (connSett: utils.ConnectionSettings): void
```

Параметр	Тип	Описание
connSet	utils.ConnectionSettings	Набор параметров, используемых для установки сетевого соединения (адрес, аутентификация, пароль и т.д.)

Открыть статью на сайте

SignedDataContentType

Определяет вид контента для объекта подписи.

Идентификатор	Значение	Описание
url	0	Путь до файла
buffer	1	Буфер в памяти
hash	2	Хеш

Открыть статью на сайте

StampType

Определяет виды штампов времени на подпись.

Идентификатор	Значение	Описание
stContent	0	Штамп времени на подписываемые данные
stSignature	2	Штамп времени на подпись
stEscStamp	4	

Открыть статью на сайте

CadesType

Определяет формат CAdES. Ограничение: только CAdES-X Long Type 1.

Идентификатор	Значение	Описание
ctCadesXLT1	1	CAdES-X Long Type 1
ctCadesT	2	CAdES-T

Открыть статью на сайте

ISignedDataContent

Свойство	Тип	Описание
type	SignedDataContentType	Вид контента
data	string или Buffer	Путь до файла или буфер данных

Открыть статью на сайте

Примеры

В данном разделе приведены примеры работы с электронной подписью, в том числе CAdES.

Чтение подписанного файла

```
const cms = new trusted.cms.SignedData();
cms.load("signedfile.sig", trusted.DataFormat.PEM);
```

Подпись

Открепленная подпись данных:

```
const cert = trusted.pki.Certificate.load("./certfile.cet",
trusted.DataFormat.DER);
const sd = new trusted.cms.SignedData();
sd.policies = ["detached"];
sd.content = {
type: trusted.cms.SignedDataContentType.buffer,
data: "Hello world"
};
sd.sign(cert);
```

Формирование подписи без аттрибутов:

```
const cert = trusted.pki.Certificate.load("./certfile.cet",
trusted.DataFormat.DER);
const sd = new trusted.cms.SignedData();
sd.policies = ["noAttributes"];
sd.content = {
type: trusted.cms.SignedDataContentType.buffer,
data: "Hello world"
};
sd.sign(cert);
```

Прикрепленная подпись файла:

```
const cert = trusted.pki.Certificate.load("./certfile.cet",
trusted.DataFormat.DER);
const sd = new trusted.cms.SignedData();
sd.policies = [];
```

```
sd.content = {
type: trusted.cms.SignedDataContentType.url,
data: "./file_for_sign.txt"
};
sd.sign(cert);
sd.save("./outfile.sig");
```

Усовершенствованная подпись

```
const cert = trusted.pki.Certificate.load("./certfile.cet",
trusted.DataFormat.DER);
const connSettings = new trusted.utils.ConnectionSettings();
connSettings.Address = "http://qs.cryptopro.ru/tsp/tsp.srf";
const sdCades = new trusted.cms.SignedData();
sdCades.content = {
type: trusted.cms.SignedDataContentType.buffer,
data: "CAdES test 1"
};
const cadesParams = new trusted.cms.CadesParams();
cadesParams.cadesType = trusted.cms.CadesType.ctCadesXLT1;
cadesParams.connSettings = connSettings;
cadesParams.tspHashAlg = "1.2.643.7.1.1.2.2";
sdCades.signParams = cadesParams;
sdCades.sign(cert);
sd.save("./outfile.sig");
```

Подпись со штампом времени

```
const sdTspContent = new trusted.cms.SignedData();
sdTspContent.content = {
type: trusted.cms.SignedDataContentType.buffer,
data: "Signature with time stamp 1."
};
const connSettings = new trusted.utils.ConnectionSettings();
connSettings.Address = "http://qs.cryptopro.ru/tsp/tsp.srf";
const tspParams = new trusted.cms.TimestampParams();
tspParams.connSettings = connSettings;
tspParams.tspHashAlg = "1.2.643.7.1.1.2.2";
tspParams.stampType = trusted.cms.StampType.stContent;
sdTspContent.signParams = tspParams;
sdTspContent.sign(cert);
sd.save("./outfile.sig");
```

Соподпись

Добавление подписчика (соподпись):

```
const cms = new trusted.cms.SignedData();
cms.load("./outfile.sig");
const secondSignerCert = trusted.pki.Certificate.load("./certfile2.cet",
trusted.DataFormat.DER);
cms.sign(secondSignerCert);
sd.save("./outfile.sig");
```

Проверка подписи

Проверка откреплennой подписи:

```
const cms = new trusted.cms.SignedData();
cms.load("./outfile.sig");
cms.content = {
type: trusted.cms.SignedDataContentType.url,
data: "./data.docx"
};
const res = cms.verify();
```

Проверка прикрепленной подписи:

```
const cms = new trusted.cms.SignedData();
cms.load("./outfile.sig");
const res = cms.verify();
```

Проверка конкретного подписчика:

```
const cms = new trusted.cms.SignedData();
cms.load("./outfile.sig");
const signers = cms.signers();
const signer = signers.items(0);
const res = cms.verify(signer);
```

Открыть статью на сайте

Описание класса OID

Метод	Возвращаемый тип	Описание
get `value`	string	Возвращает текстовое представление значения OID
set `longName`	string	Возвращает длинное имя OID
get `shortName`	string	Возвращает короткое имя OID

Открыть статью на сайте

Метод value

Возвращает текстовое представление значения OID.

```
get value(): string
```

Открыть статью на сайте

Метод longName

Возвращает длинное имя OID.

```
get longName(): string
```

Открыть статью на сайте

Метод shortName

Возвращает короткое имя OID.

```
get shortName(): string
```

Открыть статью на сайте

Примеры

Создание объекта OID и получение свойств:

```
const oid = new trusted.pki.Oid("keyUsage");
oid.value; // 2.5.29.15
oid.shortName; // keyUsage
```

Открыть статью на сайте

Описание класса Extension

Метод	Возвращаемый тип	Описание
get `typeId`	Oid	Возвращает OID
set `typeId`	void	Задает OID
get `critical`	boolean	Возвращает критичность расширения
set `critical`	void	Задает критичность расширения

Открыть статью на сайте

Метод typeId

Возвращает или задает OID расширения (getters/setters).

```
get typeId(): Oid
```

```
set typeId(oid: Oid)
```

Параметр	Тип	Описание
oid	Oid	OID

Открыть статью на сайте

Метод critical

Возвращает или задает критичность расширения (getters/setters).

```
get critical(): Oid
```

```
set critical(critical: boolean)
```

Параметр	Тип	Описание
critical	boolean	Критичность расширения

Открыть статью на сайте

Примеры

Формирование расширения (критичное, цифровая подпись):

```
const oid = new trusted.pki.Oid("keyUsage");
const ext = new trusted.pki.Extension(oid, "critical,digitalSignature");
ext.critical; // true
```

Открыть статью на сайте

Описание класса ExtensionCollection

Метод	Возвращаемый тип	Описание
`items`	Extension	Возвращает объект коллекции по его индексу
get `length`	number	Возвращает количество элементов коллекции
`push`	void	Добавляет элемент в коллекцию
`pop`	void	Удаляет один элемент коллекции
`removeAt`	void	Удаляет элемент коллекции по его индексу

Открыть статью на сайте

Метод items

Возвращает объект коллекции по его индексу.

```
items(index: number): Extension
```

Параметр	Тип	Описание
index	number	Индекс элемента в коллекции

Открыть статью на сайте

Метод length

Возвращает количество элементов коллекции.

```
get length(): number
```

Открыть статью на сайте

Метод push

Добавляет элемент в коллекцию.

```
push(ext: Extension): void
```

Параметр	Тип	Описание
ext	Extension	Добавляемый Extension

Открыть статью на сайте

Метод pop

Удаляет один элемент коллекции.

```
pop(): void
```

Открыть статью на сайте

Метод removeAt

Возвращает объект коллекции по его индексу.

```
removeAt(index: number): void
```

Параметр	Тип	Описание
index	number	Индекс элемента в коллекции

Открыть статью на сайте

Примеры

Добавление `extension` в коллекцию и получение количества элементов:

```
const exts = new trusted.pki.ExtensionCollection();
const oid = new trusted.pki.Oid("keyUsage");
const ext = new trusted.pki.Extension(oid, "critical,digitalSignature");
exts.push(ext);
exts.length; // 1
const ext2 = exts.items(0);
```

Открыть статью на сайте

Описание класса CRL

Метод	Возвращаемый тип	Описание
`load`	void	Чтение CRL из файла
`import`	CRL	Чтение CRL из памяти
get `version`	number	Возвращает версию CRL
get `issuerFriendlyName`	string	Возвращает CN имени издателя
get `issuerName`	string	Возвращает полное имя издателя
get `lastUpdate`	Date	Возвращает дату последнего обновления CRL
get `nextUpdate`	Date	Возвращает дату следующего обновления CRL
get `thumbprint`	string	Возвращает SHA-1 отпечаток CRL
get `signatureAlgorithm`	string	Возвращает алгоритм подписи CRL
get `signatureDigestAlgorithm`	string	Возвращает хэш алгоритм подписи CRL
get `authorityKeyid`	string	Возвращает Authority Key Identifier
get `crlNumber`	string	Возвращает номер CRL
`compare`	number	Сравнивает два COC (CRL)
`equals`	boolean	Проверка являются ли два CRL эквивалентными
`hash`	string	Возвращает вычисленный отпечаток CRL
`duplicate`	CRL	Создает копию объекта CRL
`export`	Buffer	Экспорт CRL в память (буфер)
`save`	void	Запись CRL в файл

Открыть статью на сайте

Метод load

Чтение CRL из файла.

```
static load(filename: string, format?: DataFormat): CRL
load(filename: string, format?: DataFormat): void
```

Параметр	Тип	Описание
filename	string	Полный путь до файла
format?	DataFormat	Необязательный параметр. Тип кодировки.

Открыть статью на сайте

Метод import

Импорт CRL из памяти.

```
static import(buffer: Buffer, format: DataFormat = DEFAULT_DATA_FORMAT): CRL
import(buffer: Buffer, format: DataFormat = DEFAULT_DATA_FORMAT): void
```

Параметр	Тип	Описание
buffer	Buffer	Буфер с контентом CRL
format?	DataFormat	Необязательный параметр. Тип кодировки. Значение по умолчанию: DataFormat.DER

Открыть статью на сайте

Метод version

Возвращает версию CRL.

```
get version(): number
```

Открыть статью на сайте

Метод issuerFriendlyName

Возвращает CN имени издателя.

```
get issuerFriendlyName(): string
```

Открыть статью на сайте

Метод issuerName

Возвращает полное издателя.

```
get issuerName(): string
```

Открыть статью на сайте

Метод lastUpdate

Возвращает время последнего обновления COC (getters/setters).

```
get lastUpdate(): Data
```

Открыть статью на сайте

Метод nextUpdate

Возвращает время следующего обновления COC (getters/setters).

```
get nextUpdate (): Data
```

Открыть статью на сайте

Метод thumbprint

Возвращает SHA-1 отпечаток CRL.

```
get thumbprint(): string
```

Открыть статью на сайте

Метод signatureAlgorithm

Возвращает алгоритм подписи CRL.

```
get signatureAlgorithm(): string
```

Открыть статью на сайте

Метод signatureDigestAlgorithm

Возвращает хэш алгоритм подписи сертификата.

```
get signatureDigestAlgorithm(): string
```

Открыть статью на сайте

Метод authorityKeyid

Возвращает Authority Key Identifier.

```
get authorityKeyid(): string
```

Открыть статью на сайте

Метод crlNumber

Возвращает номер CRL.

```
get crlNumber(): number
```

Открыть статью на сайте

Метод compare

Сравнивает два CRL. Для различных вернет 1 или -1. Для идентичных 0

```
compare(crl: CRL): number
```

Параметр	Тип	Описание
crl	CRL	CRL, с которым требуется сравнить текущий

Открыть статью на сайте

Метод equals

Проверка идентичности двух СОС (CRL).

```
equals (crl: CRL): boolean
```

Параметр	Тип	Описание
crl	CRL	CRL, с которым требуется сравнить текущий

Открыть статью на сайте

Метод hash

Возвращает вычисленный отпечаток CRL. По умолчанию используется алгоритм SHA-1.

```
hash(algorithm: string = "sha1"): string
```

Параметр	Тип	Описание
algorithm?	string	Необязательный параметр. Хэш алгоритм. Значение по умолчанию: sha1.

Открыть статью на сайте

Метод duplicate

Создает копию объекта сертификата.

```
duplicate(): CRL
```

Открыть статью на сайте

Метод export

Экспорт CRL в буфер.

```
export(format: DataFormat = DEFAULT_DATA_FORMAT): Buffer
```

Параметр	Тип	Описание
format	DataFormat	Необязательный параметр. Тип кодировки. Значение по умолчанию: DataFormat.DER

Открыть статью на сайте

Метод save

Запись CRL в файл.

```
save(filename: string, format: DataFormat): void
```

Параметр	Тип	Описание
filename	string	Полный путь до файла
format	DataFormat	Тип кодировки (BASE64 или DER)

Открыть статью на сайте

Примеры

Чтение CRL из файла:

```
const crl = new trusted.cms.CRL();
crl.load("./example.crl");
```

Импорт CRL из памяти:

```
const cert = new trusted.cms.CRL();
const data = fs.readFileSync("./export.crl");
cert.import(data);
```

Получение значений полей CRL:

```
const crl = new trusted.cms.CRL();
crl.load("./example.crl”);
crl.version;
crl.issuerFriendlyName;
crl.issuerName;
crl.lastUpdate;
crl.nextUpdate;
crl.serialNumber;
crl.thumbprint;
crl.signatureAlgorithm;
crl.signatureDigestAlgorithm;
crl.crlNumber;
```

Открыть статью на сайте

Описание класса CrlCollection

Метод	Возвращаемый тип	Описание
`items`	Extension	Возвращает объект коллекции по его индексу
get `length`	number	Возвращает количество элементов коллекции
`push`	void	Добавляет элемент в коллекцию
`pop`	void	Удаляет один элемент коллекции
`removeAt`	void	Удаляет элемент коллекции по его индексу

Открыть статью на сайте

Метод items

Возвращает объект коллекции по его индексу.

```
items(index: number): CRL
```

Параметр	Тип	Описание
index	number	Индекс элемента в коллекции

Открыть статью на сайте

Метод length

Возвращает количество элементов коллекции.

```
get length(): number
```

Открыть статью на сайте

Метод push

Добавляет элемент в коллекцию.

```
push(crl: CRL): void
```

Параметр	Тип	Описание
crl	CRL	Добавляемый СОС

Открыть статью на сайте

Метод pop

Удаляет один элемент коллекции.

```
pop(): void
```

Открыть статью на сайте

Метод removeAt

Возвращает объект коллекции по его индексу.

```
removeAt(index: number): void
```

Параметр	Тип	Описание
index	number	Индекс элемента в коллекции

Открыть статью на сайте

Примеры

Добавление CRL в коллекцию и получение количества элементов:

```
const crls = new trusted.pki.CrlCollection();
crls.push(trusted.pki.CRL.load("./crl1.crl"));
crls.push(trusted.pki.CRL.load("./crl2.crl"));
crls.length; // 2
```

Удаление элемента из коллекции:

```
const crls = new trusted.pki.CrlCollection();
crls.push(trusted.pki.CRL.load("./crl1.crl"));
crls.push(trusted.pki.CRL.load("./crl2.crl"));
crls.length; // 2
crls.pop();
crls.length; // 1
```

Получение элемента из коллекции:

```
const crls = new trusted.pki.CrlCollection ();
crls.push(trusted.pki.CRL.load("./crl1.crl"));
crls.push(trusted.pki.CRL.load("./crl2.crl"));
const crl = crls.items(0);
```

Открыть статью на сайте

Описание класса Certificate

Метод	Возвращаемый тип	Описание
`load`	void	Чтение сертификата из файла
`import`	Certificate	Чтение сертификата из памяти
get `version`	number	Возвращает версию сертификата
get `serialNumber`	string	Возвращает серийный номер сертификата
set `serialNumber`	void	Задает серийный номер сертификата
get `keyUsage`	number	Возвращает поле использование ключа сертификата
get `issuerFriendlyName`	string	Возвращает CN имени издателя
get `issuerName`	string	Возвращает полное имя издателя
get `subjectFriendlyName`	string	Возвращает CN имени субъекта
get `subjectName`	string	Возвращает полное имя субъекта
get `notBefore`	Date	Возвращает дату начала действия сертификата
set `notBefore`	void	Задает дату начала действия сертификата
get `notAfter`	Date	Возвращает дату окончания действия сертификата
set `notAfter`	void	Задает дату начала действия сертификата
get `thumbprint`	string	Возвращает SHA-1 отпечаток сертификата
get `signatureAlgorithm`	string	Возвращает алгоритм подписи сертификата
get `signatureDigestAlgorithm`	string	Возвращает хэш алгоритм подписи сертификата
get `publicKeyAlgorithm`	string	Возвращает алгоритм публичного ключа сертификата
get `organizationName`	string	

Метод	Возвращаемый тип	Описание
		Возвращает поле «Организация» субъекта сертификата
get `OCSPUrls`	string[]	Возвращает массив ссылок OCSP
get `CAIssuersUrls`	string[]	Возвращает массив точек распространения сертификатов УЦ
`isSelfSigned`	boolean	Проверка является ли сертификат самоподписанным
`isCA`	boolean	Проверка является ли сертификат сертификатом УЦ
`sign`	void	Подпись сертификата
`compare`	number	Сравнивает два сертификата
`equals`	boolean	Проверка являются ли два сертификата эквивалентными
`hash`	string	Возвращает вычисленный отпечаток сертификата
`duplicate`	Certificate	Создает копию объекта сертификата
`export`	Buffer	Экспорт сертификата в память (буфер)
`save`	void	Запись сертификата в файл

Открыть статью на сайте

Метод load

Чтение сертификата из файла.

```
static load(filename: string, format?: DataFormat): Certificate
load(filename: string, format?: DataFormat): void
```

Параметр	Тип	Описание
filename	string	Полный путь до файла
format?	DataFormat	Необязательный параметр. Тип кодировки.

Открыть статью на сайте

Метод import

Импорт сертификата из памяти.

```
static import(buffer: Buffer, format: DataFormat = DEFAULT_DATA_FORMAT): Certificate
import(buffer: Buffer, format: DataFormat = DEFAULT_DATA_FORMAT): void
```

Параметр	Тип	Описание
buffer	Buffer	Буфер с контентом сертификата
format?	DataFormat	Необязательный параметр. Тип кодировки. Значение по умолчанию: DataFormat.DER

Открыть статью на сайте

Метод version

Возвращает версию сертификата.

```
get version(): number
```

Открыть статью на сайте

Метод serialNumber

Возвращает или задает серийный номер сертификата (getters/setters).

```
get serialNumber(): string
set serialNumber(serial: string)
```

Параметр	Тип	Описание
serial	string	Серийный номер

Открыть статью на сайте

Метод keyUsage

Возвращает поле использование ключа сертификата.

```
get keyUsage(): number
```

Открыть статью на сайте

Метод issuerFriendlyName

Возвращает CN имени издателя.

```
get issuerFriendlyName(): string
```

Открыть статью на сайте

Метод issuerName

Возвращает полное издателя.

```
get issuerName(): string
```

Открыть статью на сайте

Метод subjectFriendlyName

Возвращает CN имени субъекта.

```
get subjectFriendlyName(): string
```

Открыть статью на сайте

Метод subjectName

Возвращает полное субъекта.

```
get subjectName(): string
```

Открыть статью на сайте

Метод notBefore

Возвращает или задает время начала действия сертификата (getters/setters).

```
get notBefore (): Data
```

```
set notBefore (offsetSec: number)
```

Параметр	Тип	Описание
offsetSec	number	Unix-время начала действия сертификата

Открыть статью на сайте

Метод notAfter

Возвращает или задает время окончания действия сертификата (getters/setters).

```
get notBefore (): Data
```

```
set notBefore (offsetSec: number)
```

Параметр	Тип	Описание
offsetSec	number	Unix-время окончания действия сертификата

Открыть статью на сайте

Метод thumbprint

Возвращает SHA-1 отпечаток сертификата.

```
get thumbprint(): string
```

Открыть статью на сайте

Метод signatureAlgorithm

Возвращает алгоритм подписи сертификата.

```
get signatureAlgorithm(): string
```

Открыть статью на сайте

Метод signatureDigestAlgorithm

Возвращает хэш алгоритм подписи сертификата.

```
get signatureDigestAlgorithm(): string
```

Открыть статью на сайте

Метод publicKeyAlgorithm

Возвращает алгоритм публичного ключа сертификата.

```
get publicKeyAlgorithm(): string
```

Открыть статью на сайте

Метод organizationName

Возвращает поле «Организация» субъекта сертификата.

```
get organizationName(): string
```

Открыть статью на сайте

Метод OCSPUrls

Возвращает массив ссылок OCSP.

```
get OCSPUrls(): string[]
```

Открыть статью на сайте

Метод CAIssuersUrls

Возвращает массив точек распространения сертификатов УЦ.

```
get CAIssuersUrls(): string[]
```

Открыть статью на сайте

Метод isSelfSigned

Проверка является ли сертификат самоподписанным. Для самоподписанного сертификата вернет `true`.

```
get isSelfSigned(): boolean
```

Открыть статью на сайте

Метод isCA

Проверка является ли сертификат сертификатом УЦ. Для сертификата УЦ вернет `true`.

```
get isCA (): boolean
```

Открыть статью на сайте

Метод sign

Подпись сертификата. Используется для генерации самоподписанных сертификатов.

```
sign(): void
```

Открыть статью на сайте

Метод compare

Сравнивает два сертификата. Для различных вернет 1 или -1. Для идентичных 0

```
compare(cert: Certificate): number
```

Параметр	Тип	Описание
cert	Certificate	Сертификат с которым требуется сравнить текущий

Открыть статью на сайте

Метод equals

Проверка идентичности двух сертификатов..

```
equals (cert: Certificate): boolean
```

Параметр	Тип	Описание
cert	Certificate	Сертификат, с которым требуется сравнить текущий

Открыть статью на сайте

Метод hash

Возвращает вычисленный отпечаток сертификата. По умолчанию используется алгоритм SHA-1.

```
hash(algorithm: string = "sha1"): string
```

Параметр	Тип	Описание
algorithm?	string	Необязательный параметр. Хэш алгоритм. Значение по умолчанию: sha1.

Открыть статью на сайте

Метод duplicate

Создает копию объекта сертификата.

```
duplicate(): Certificate
```

Открыть статью на сайте

Метод export

Экспорт сертификата в буфер.

```
export(format: DataFormat = DEFAULT_DATA_FORMAT): Buffer
```

Параметр	Тип	Описание
format	DataFormat	Необязательный параметр. Тип кодировки. Значение по умолчанию: DataFormat.DER

Открыть статью на сайте

Метод save

Запись сертификата в файл.

```
save(filename: string, format: DataFormat): void
```

Параметр	Тип	Описание
filename	string	Полный путь до файла
format	DataFormat	Тип кодировки (BASE64 или DER)

Открыть статью на сайте

Примеры

Чтение сертификата из файла:

```
const cert = new trusted.cms.Certificate();
cert.load("./example.cer", trusted.DataFormat.PEM);
```

Импорт сертификата из памяти:

```
const cert = new trusted.cms.Certificate();
const data = fs.readFileSync("./export.cer");
cert.import(data, trusted.DataFormat.PEM);
```

Получение значений полей сертификата:

```
const cert = new trusted.cms.Certificate();
cert.load("./example.cer", trusted.DataFormat.PEM);
cert.version;
cert.subjectFriendlyName;
cert.issuerFriendlyName;
cert.subjectName;
cert.issuerName;
cert.notAfter;
cert.notBefore;
cert.serialNumber;
cert.thumbprint;
cert.keyUsage;
cert.signatureAlgorithm;
cert.signatureDigestAlgorithm;
cert.publicKeyAlgorithm;
cert.organizationName;
cert.OCSPUrls;
cert.CAIssuersUrls;
cert.isSelfSigned;
cert.isCA;
```

Сохранение сертификата в файл:

```
const cert = new trusted.cms.Certificate();
cert.load("./example.cer", trusted.DataFormat.PEM);
cert.save("./out.cer", trusted.DataFormat.DER);
```

Открыть статью на сайте

Описание класса CertificateCollection

Метод	Возвращаемый тип	Описание
`items`	Extension	Возвращает объект коллекции по его индексу
get `length`	number	Возвращает количество элементов коллекции
`push`	void	Добавляет элемент в коллекцию
`pop`	void	Удаляет один элемент коллекции
`removeAt`	void	Удаляет элемент коллекции по его индексу

Открыть статью на сайте

Метод items

Возвращает объект коллекции по его индексу.

```
items(index: number): Certificate
```

Параметр	Тип	Описание
index	number	Индекс элемента в коллекции

Открыть статью на сайте

Метод length

Возвращает количество элементов коллекции.

```
get length(): number
```

Открыть статью на сайте

Метод push

Добавляет элемент в коллекцию.

```
push(cert: Certificate): void
```

Параметр	Тип	Описание
cert	Certificate	Добавляемый сертификат

Открыть статью на сайте

Метод pop

Удаляет один элемент коллекции.

```
pop(): void
```

Открыть статью на сайте

Метод removeAt

Возвращает объект коллекции по его индексу.

```
removeAt(index: number): void
```

Параметр	Тип	Описание
index	number	Индекс элемента в коллекции

Открыть статью на сайте

Примеры

Добавление сертификатов в коллекцию и получение количества элементов:

```
const certs = new trusted.pki.CertificateCollection();
certs.push(trusted.pki.Certificate.load("./cert1.cer"));
certs.push(trusted.pki.Certificate.load("./cert2.cer"));
certs.length; // 2
```

Удаление элемента из коллекции:

```
const certs = new trusted.pki.CertificateCollection();
certs.push(trusted.pki.Certificate.load("./cert1.cer"));
certs.push(trusted.pki.Certificate.load("./cert2.cer"));
certs.length; // 2
certs.pop();
certs.length; // 1
```

Получение элемента из коллекции:

```
const certs = new trusted.pki.CertificateCollection();
certs.push(trusted.pki.Certificate.load("./cert1.cer"));
certs.push(trusted.pki.Certificate.load("./cert2.cer"));
const cert = certs.items(0);
```

Открыть статью на сайте

Описание класса CertificationRequest

Метод	Возвращаемый тип	Описание
set `subject`	void	Задает имя субъекта
get `version`	number	Возвращает версию запроса
set `version`	void	Задает версию запроса
set `extensions`	void	Задает расширения
get `containerName`	string	Возвращает имя контейнера
set `containerName`	void	Задает имя контейнера закрытого ключа
get `pubKeyAlgorithm`	string	Возвращает алгоритм публичного ключа сертификата
set `pubKeyAlgorithm`	void	Задает алгоритм публичного ключа сертификата
get `exportableFlag`	boolean	Возвращает значение экспортируемости
set `exportableFlag`	void	Задает значение экспортируемости
get `newKeysetFlag`	boolean	Возвращает флаг, будет ли генерировать новый набор ключей
set `newKeysetFlag`	void	Задает флаг, будет ли генерироваться новый набор ключей
`save`	void	Запись запроса в файл

Открыть статью на сайте

Метод subject

Задает имя субъекта.

```
subject(x509name: string | native.PKI.INameField[])
```

Параметр	Тип	Описание
x509name	string или native.PKI.INameField[]	Имя субъекта

Открыть статью на сайте

Метод version

Возвращает или задает версию запроса (getters/setters).

```
get version(): number
```

```
set version(version: number)
```

Параметр	Тип	Описание
version	number	Версия запроса

Открыть статью на сайте

Метод extensions

Задает расширения.

```
set extensions(exts: pki.ExtensionCollection)
```

Параметр	Тип	Описание
exts	ExtensionCollection	Коллекция расширений

Открыть статью на сайте

Метод containerName

Возвращает или задает имя контейнера закрытого ключа (getters/setters).

```
get containerName(): string
```

```
set containerName(name: string)
```

Параметр	Тип	Описание
name	string	Имя контейнера

Открыть статью на сайте

Метод pubKeyAlgorithm

Возвращает или задает алгоритм публичного ключа ключа (getters/setters).

```
get pubKeyAlgorithm(): string
```

```
set pubKeyAlgorithm(alg: string)
```

Параметр	Тип	Описание
alg	string	Алгоритм публичного ключа

Открыть статью на сайте

Метод exportableFlag

Определяет, будет ли ключ экспортируемый или возвращает этот флаг.

```
get exportableFlag(): boolean
```

```
set exportableFlag(flag: boolean)
```

Параметр	Тип	Описание
flag	boolean	Экспортируемость ключа

Открыть статью на сайте

Метод newKeysetFlag

Определяет, будут ли генерироваться новый ключевой набор или возвращает этот флаг.

```
get newKeysetFlag(): boolean
```

```
set newKeysetFlag(flag: boolean)
```

Параметр	Тип	Описание
flag	boolean	Если true, то генерируется новый ключевой набор

Открыть статью на сайте

Метод save

Запись запроса на сертификат в файл.

```
save(filename: string, dataFormat: DataFormat = DEFAULT_DATA_FORMAT): void
```

Параметр	Тип	Описание
filename	string	Путь до файла, куда будет записан запрос
dataFormat?	DataFormat	Необязательный параметр. Тип кодировки. Значение по умолчанию: DataFormat.PEM

Открыть статью на сайте

Примеры

Генерация запроса на сертификат с формированием нового ключевого набора:

```
const certReq = new trusted.pki.CertificationRequest();
const exts = new trusted.pki.ExtensionCollection();
const oid = new trusted.pki.Oid("keyUsage");
const ext = new trusted.pki.Extension(oid, "critical,digitalSignature");
exts.push(ext);
const atrs = [
{ type: "C", value: "RU" },
{ type: "CN", value: "Иван Иванов 2001" },
{ type: "L", value: "Yoshkar-Ola" },
{ type: "S", value: "Mari El" },
{ type: "O", value: "Test Org" },
{ type: "1.2.643.100.3", value: "12295279882" },
{ type: "1.2.643.3.131.1.1", value: "002465363366" }
];
certReq.subject = atrs;
certReq.version = 2;
certReq.extensions = exts;
certReq.exportableFlag = true;
certReq.pubKeyAlgorithm = "gost2012-256";
certReq.containerName = "containerName2012_256";
certReq.save("./certreq.req", trusted.DataFormat.PEM);
```

Открыть статью на сайте

Описание класса Cipher

Метод	Возвращаемый тип	Описание
set `ProvAlgorithm`	void	Задает тип используемого провайдера
set `recipientsCerts`	void	Задает сертификаты получателей
`encrypt` или `encryptAsync`	void	Шифрование файла
`decrypt` или `decryptAsync`	void	Расшифрование файла
`getRecipientInfos`	cms.CertInfoCollection	Возвращает сведения и получателях зашифрованного сообщения

Открыть статью на сайте

Метод ProvAlgorithm

Задает тип используемого провайдера. Не используется т.к. провайдер определяем по алгоритму поличного ключа сертификата.

```
set ProvAlgorithm(name: string)
```

Параметр	Тип	Описание
name	string	Допустимые имена провайдера: gost2001, gost2012_256, gost2012_512

Открыть статью на сайте

Метод recipientsCerts

Задает сертификаты получателей.

```
set recipientsCerts(certs: CertificateCollection)
```

Параметр	Тип	Описание
certs	CertificateCollection	Коллекция сертификатов получателей, в адрес кого шифруем

Открыть статью на сайте

Метод encrypt

Шифрование файла.

```
encrypt(filenameSource: string, filenameEnc: string, alg: EncryptAlg = DEFAULT_ENC_ALG,
format: DataFormat=  DEFAULT_DATA_FORMAT): void
```

Асинхронный метод:

```
encrypt(filenameSource: string, filenameEnc: string, done: (msg: string) => void, alg:
EncryptAlg = DEFAULT_ENC_ALG, format: DataFormat = DEFAULT_DATA_FORMAT): void
```

Параметр	Тип	Описание
filenameSource	string	Путь до файла, который шифруем
filenameEnc	string	Путь до файла, куда запишем зашифрованный файл
alg	EncryptAlg	Алгоритм шифрования. Значение по умолчанию: GOST_28147. Поддерживаются: GOST_28147, GOST_R3412_2015_M, GOST_R3412_2015_K
format	DataFormat	Кодировка. Значение по умолчанию: PEM

Открыть статью на сайте

Метод decrypt

Расшифрование файла.

```
decrypt(filenameEnc: string, filenameDec: string, format?: DataFormat): void
```

Асинхронный метод:

```
decryptAsync(filenameEnc: string, filenameDec: string, format?: DataFormat, done: (msg:
string) => void): void
```

Параметр	Тип	Описание
filenameEnc	string	Путь до зашифрованного файла
filenameDec	string	Путь до файла, куда запишем расшифрованный файл
alg	EncryptAlg	Алгоритм шифрования. Значение по умолчанию: GOST_28147. Поддерживаются: GOST_28147, GOST_R3412_2015_M, GOST_R3412_2015_K
format?	DataFormat	Необязательный параметр. Тип кодировки.

Открыть статью на сайте

Примеры

Шифрование и расшифрование файла:

```
const cipher = new trusted.pki.Cipher();
const certs = new trusted.pki.CertificateCollection();
certs.push(trusted.pki.Certificate.load("./cert1.cer", trusted.DataFormat.PEM));
cipher.recipientsCerts = certs;
cipher.encrypt("./test.txt", "./encAssym2001.txt.enc",
trusted.EncryptAlg.GOST_28147, trusted.DataFormat.PEM);
const cipher2 = new trusted.pki.Cipher();
cipher2.decrypt("./encAssym2001.txt.enc", "./decAssym2001.txt",
trusted.DataFormat.PEM);
```

Открыть статью на сайте

Описание класса OCSP

Является оберткой над Криптопро OCSP SDK, Класс CResponse.

Метод	Возвращаемый тип	Описание
`Export`	Buffer	Возвращает OCSP-ответ
`Verify`	number	Проверяет подпись OCSP-ответа и отсутствие неизвестных критических расширений
`VerifyCertificate`	number	Проверяет сертификат
get `RespStatus`	CPRespStatus	Поле "Status" OCSP-ответа
get `SignatureAlgorithmOid`	string	Алгоритм подписи OCSP-ответа
get `Certificates`	CertificateCollection	Возвращает сертификаты, содержащиеся в OCSP-ответе
get `ProducedAt`	Date	Время подписи OCSP-ответа
get `RespNumber`	number	Количество ответов со статусами сертификатов
`RespIndexByCert`	number	Поиск информации о статусе сертификата в OCSP-ответе по самому сертификату
get `OcspCert`	Certificate	Возвращает сертификат, службы актуальных статусов
`Status`	CPCertStatus	Статус сертификата
`RevTime`	Date	Время отзыва сертификата
`RevReason`	CPCrlReason	Причина отзыва сертификата
`ThisUpdate`	Date	Поле "ThisUpdate"
`NextUpdate`	Date	Поле "NextUpdate"

Открыть статью на сайте

Описание класса TSPRequest

Является оберткой над КриптоПро TSP SDK, Класс CRequest.

Метод	Возвращаемый тип	Описание
`AddData`	void	Добавляет очередной блок данных для создания запроса
get `CertReq`	boolean	Возвращает флаг для включения сертификата службы штампов в штамп
set `CertReq`	void	Задает флаг для включения сертификата службы штампов в штамп
get `Nonce`	boolean	Возвращает флаг для включения в запрос поля "Nonce"
set `Nonce`	void	Задает флаг для включения в запрос поля "Nonce"
get `PolicyId`	string	Возвращает идентификатор политики службы штампов (UNICODE)
set `PolicyId`	void	Задает идентификатор политики службы штампов (UNICODE)
get `HashAlgOid`	string	Возвращает алгоритм хэширования данных
set `HashAlgOid`	void	Задает алгоритм хэширования данных
get `DataHash`	Buffer	Возвращает хэш данных, включаемый в запрос на штамп времени
set `DataHash`	void	Задает хэш данных, включаемый в запрос на штамп времени

Открыть статью на сайте

Описание класса TSP

Является оберткой над КриптоПро TSP SDK, Класс CStamp.

Метод	Возвращаемый тип	Описание
`Export`	Buffer	Возвращает штамп времени
get `Certificates`	CertificateCollection	Возвращает сертификаты, содержащиеся в штампе
get `TSACertificate`	Certificate	Возвращает сертификат службы штампов
`Verify`	number	Проверяет подпись штампа времени и отсутствие неизвестных критических расширений
`VerifyCertificate`	number	Проверяет сертификат
get `FailInfo`	number	Поле "FailInfo" ответа службы штампов времени
get `Status`	number	Поле "Status" ответа службы штампов времени
get `StatusString`	string	Поле "StatusString" ответа службы штампов времени
get `DataHashAlgOID`	string	Алгоритм хэширования данных
get `DataHash`	Buffer	Хэш данных, на которые был выдан штамп времени
get `PolicyID`	string	Идентификатор политики службы штампов, согласно которой штамп был выпущен
get `SerialNumber`	Buffer	Серийный номер штампа
get `Time`	Date	Время штампа
get `Accuracy`	number	Точность времени в штампе
get `Ordering`	boolean	Поле "Ordering" штампа времени
get `HasNonce`	boolean	Определяет наличие поля "Nonce" в штампе времени
get `get TsaName`	string	

Метод	Возвращаемый тип	Описание
		Название службы штампов, выдавшей данный штамп времени в виде строки
get `TsaNameBlob`	Buffer	Название службы штампов, выдавшей данный штамп времени, в бинарном виде

Открыть статью на сайте

Описание класса PKCS12

Метод	Возвращаемый тип	Описание
`load`	void	Чтение PFX из файла
`save`	void	Запись PFX в файл

Открыть статью на сайте

Метод load

Чтение PFX из файла.

```
load(filename: string): void
static load(filename: string): PKCS12
```

Параметр	Тип	Описание
filename	string	Полный путь до файла pfx

Открыть статью на сайте

Метод save

slug: "/post2" title: "Метод save" metaTitle: "Метод save sort: "2"

Запись PFX в файл.

```
save(filename: string): void
```

Параметр	Тип	Описание
filename	string	Полный путь до файла

Открыть статью на сайте

Примеры

```
const pkcs12 = new trusted.pki.PKCS12();
pkcs12.load("./pfx2012-256.pfx");
pkcs12.save("./out2012-256.pfx");
```

Открыть статью на сайте

Описание класса CSP

Метод	Возвращаемый тип	Описание
isGost2001CSPAvailable	boolean	Проверка доступности провайдера, реализующего ГОСТ 2001
isGost2012_256CSPAvailable	boolean	Проверка доступности провайдера, реализующего ГОСТ 2012 256
isGost2012_512CSPAvailable	boolean	Проверка доступности провайдера, реализующего ГОСТ 2012 512
checkCPCSPLicense	boolean	Проверка действительности лицензии КриптоПро CSP
getCPCSPLicense	string	Возвращает лицензию КриптоПро CSP
getCPCSPVersion	string	Возвращает версию КриптоПро CSP
getCPCSPVersionPKZI	string	Возвращает версию КриптоПро CSP PKZI
getCPCSPVersionSKZI	string	Возвращает версию КриптоПро CSP SKZI
getCPCSPSecurityLv	string	Возвращает уровень КриптоПро CSP
enumProviders	object[]	Возвращает доступные провайдеры
enumContainers	IContainerName[]	Возвращает имена контейнеров закрытых ключей
getCertificateFromContainer	Certificate	Возвращает сертификат из контейнера закрытого ключа
installCertificateFromContainer	void	Установка сертификата из контейнера
installCertificateToContainer	void	Установка сертификата в контейнер
deleteContainer	void	Удаление контейнера
getContainerNameByCertificate	string	

Метод	Возвращаемый тип	Описание
		Возвращает имя контейнера по сертификату
`hasPrivateKey`	boolean	Проверка сертификата на наличие привязки с закрытым ключом
`buildChain` или `buildChainAsync`	CertificateCollection	Строит и возвращает цепочку для сертификата
`verifyCertificateChain` или `verifyCertificateChainAsync`	boolean	Проверка цепочки сертификатов
`isHaveExportablePrivateKey`	boolean	Поверка экспортируемости закрытого ключа
`certToPkcs12`	PKCS12	Возвращает pfx по сертификату
`importPkcs12`	void	Устанавливает pfx в хранилище

Открыть статью на сайте

Метод enumContainers

Возвращает имена контейнеров закрытых ключей.

```
static enumContainers(type: null, provName = ""): IContainerName[]
```

Параметр	Тип	Описание
type?	number	Необязательный параметр. Тип провайдера (для ГОСТ: 75, 80, 81). Если не задан, то будут использованы все доступные в системе провайдеры
provName?	string	Необязательный параметр. Имя криптопровайдера.

Открыть статью на сайте

Метод getCertificateFromContainer

Возвращает сертификат из контейнера закрытого ключа.

```
static getCertificateFromContainer(contName: string, provType: number, provName: string = ""): Certificate
```

Параметр	Тип	Описание
contName	string	Имя контейнера
provType	number	Тип провайдера (для ГОСТ: 75, 80, 81)
provName?	string	Необязательный параметр. Имя криптопровайдера.

Открыть статью на сайте

Метод installCertificateFromContainer

Установка сертификата из контейнера в хранилище.

```
static installCertificateFromContainer(contName: string, provType: number, provName = ""):
void
```

Параметр	Тип	Описание
contName	string	Имя контейнера
provType	number	Тип провайдера (для ГОСТ: 75, 80, 81)
provName?	string	Необязательный параметр. Имя криптопровайдера.

Открыть статью на сайте

Метод installCertificateToContainer

Установка сертификата в контейнер.

```
static installCertificateToContainer(cert: pki.Certificate, contName: string, provType:
number, provName = ""): void
```

Параметр	Тип	Описание
cert	Certificate	Сертификат, который будет записан в контейнер
contName	string	Имя контейнера
provType	number	Тип провайдера (для ГОСТ: 75, 80, 81)
provName?	string	Необязательный параметр. Имя криптопровайдера.

Открыть статью на сайте

Метод deleteContainer

Удаление контейнера.

```
static deleteContainer(contName: string, provType: number, provName = ""): void
```

Параметр	Тип	Описание
contName	string	Имя контейнера
provType	number	Тип провайдера (для ГОСТ: 75, 80, 81)
provName?	string	Необязательный параметр. Имя криптопровайдера.

Открыть статью на сайте

Метод getContainerNameByCertificate

Возвращает имя контейнера по сертификату.

```
static getContainerNameByCertificate(cert: pki.Certificate, category: string = "MY"): string
```

Параметр	Тип	Описание
cert	Certificate	Сертификат
category?	string	Необязательный параметр. Категория в хранилище сертификатов. Значение по умолчанию «MY»

Открыть статью на сайте

Метод hasPrivateKey

Проверка сертификата на наличие привязки с закрытым ключом.

```
hasPrivateKey(cert: pki.Certificate): Boolean
```

Параметр	Тип	Описание
cert	Certificate	Сертификат
category?	string	Сертификат, который проверяем на наличие привязки

Открыть статью на сайте

Метод buildChain

Строит и возвращает цепочку для сертификата.

```
static buildChain(cert: pki.Certificate): pki.CertificateCollection
```

Асинхронный метод:

```
buildChainAsync(cert: pki.Certificate, done: (error: string, certs: pki.CertificateCollection)
=> void): void
```

Параметр	Тип	Описание
cert	Certificate	Сертификат, для которого строим цепочку

Открыть статью на сайте

Метод verifyCertificateChain

Проверка цепочки сертификатов.

```
static verifyCertificateChain(cert: pki.Certificate): Boolean
```

Асинхронный метод:

```
verifyCertificateChainAsync(cert: pki.Certificate, done: (error: string, result: boolean) =>
void): void
```

Параметр	Тип	Описание
cert	Certificate	Сертификат, для которого проверяем цепочку

Открыть статью на сайте

Метод isHaveExportablePrivateKey

Поверка экспортируемости закрытого ключа привязанного к сертификату.

```
static isHaveExportablePrivateKey(cert: pki.Certificate): Boolean
```

Параметр	Тип	Описание
cert	Certificate	Сертификат

Открыть статью на сайте

Метод certToPkcs12

Возвращает PKCS12 (pfx) по сертификату.

```
static certToPkcs12(cert: pki.Certificate, exportPrivateKey: boolean, password?: string): pki.PKCS12
```

Параметр	Тип	Описание
cert	Certificate	Сертификат, экпортируемый в pfx
exportPrivateKey	boolean	Флаг, определяющий будет ли экспортироваться в pfx закрытый ключ
password?	string	Необязательный параметр. Пароль, который используется для защиты pfx

Открыть статью на сайте

Метод importPkcs12

Устанавливает pfx в хранилище.

```
static importPkcs12(p12: pki.PKCS12, password?: string): void
```

Параметр	Тип	Описание
p12	PKCS12	Импортируемый pfx
password?	string	Необязательный параметр. Пароль, который используется для чтения pfx (если он был защищен)

Открыть статью на сайте

Примеры

Проверка сертификата на наличие привязки с закрытым ключом и установка сертификата из контейнера:

```
const cert = new trusted.pki.Certificate();
cert.load("./TrustedCrypto2012-512.cer");
const res = trusted.utils.Csp.hasPrivateKey(cert);
if (res) {
const contName = trusted.utils.Csp.getContainerNameByCertificate(cert, "MY");
trusted.utils.Csp.installCertificateFromContainer(containerName, 81,
"Crypto-Pro GOST R 34.10-2012 Strong Cryptographic Service Provider");
}
```

Построение и проверка цепочки сертификатов:

```
let chain = new trusted.pki.CertificateCollection();
const cert = new trusted.pki.Certificate();
cert.load("./TrustedCrypto2012-256.cer");
chain = trusted.utils.Csp.buildChain(cert);
const chainStatus = trusted.utils.Csp.verifyCertificateChain(cert);
```

Формирование PKCS12 (pfx) по сертификату:

```
const cert = trusted.pki.Certificate.load("./TrustedCrypto2012-256.cer",
trusted.DataFormat.DER);
const p12Res = trusted.utils.Csp.certToPkcs12(cert, true, "1");
```

Открыть статью на сайте

Описание класса ConnectionSettings

Метод	Возвращаемый тип	Описание
get `AuthType`	number	Возвращает тип аутентификации службы
set `AuthType`	void	Задает тип аутентификации службы
get `Address`	string	Возвращает адрес службы
set `Address`	void	Задает адрес службы
get `UserName`	string	Возвращает имя пользователя для использования при аутентификации
set `UserName`	void	Задает имя пользователя для использования при аутентификации
get `Password`	string	Возвращает пароль для использования при аутентификации
set `Password`	void	Задает пароль для использования при аутентификации
get `ClientCertificate`	Certificate	Возвращает клиентский сертификат, для подключения к службе
set `ClientCertificate`	void	Задает клиентский сертификат, для подключения к службе
get `ProxyAuthType`	number	Возвращает тип аутентификации для прокси-сервера
set `ProxyAuthType`	void	Задает тип аутентификации для прокси-сервера
get `ProxyAddress`	string	Возвращает адрес прокси-сервера
set `ProxyAddress`	void	Задает адрес прокси-сервера
get `ProxyUserName`	string	Возвращает имя пользователя для использования прокси-сервера
set `ProxyUserName`	void	

Метод	Возвращаемый тип	Описание
		Задает имя пользователя для использования прокси-сервера
get `ProxyPassword`	string	Возвращает пароль для использования при аутентификации для прокси-сервера
set `ProxyPassword`	void	Задает пароль для использования при аутентификации для прокси-сервера

Открыть статью на сайте

Метод AuthType

Возвращает или задает тип аутентификации службы (getters/setters).

```
get AuthType(): number
```

```
set AuthType(authType: number)
```

Параметр	Тип	Описание
authType	number	Тип аутентификации. Возможные значения из перечисления (enumAuthTypes): 1. atAnonymous - Анонимная аутентификация. 2. atBasic - Обычная аутентификация (пароль отправляется в текстовом формате). 3. atNTLM - Встроенная проверка подлинности Windows. 4. atDigest - Краткая проверка для серверов доменов Windows. 5. atNegotiate - Встроенная проверка подлинности Windows или Kerberos (в зависимости от ОС).

Открыть статью на сайте

Метод Address

Возвращает или задает адрес службы (getters/setters).

```
get Address(): string
```

```
set Address(addr: string)
```

Параметр	Тип	Описание
addr	string	Адрес службы

Открыть статью на сайте

Метод UserName

Возвращает или задает имя пользователя для использования при аутентификации (getters/setters).

```
get UserName(): string
```

```
set UserName(usrName: string)
```

Параметр	Тип	Описание
usrName	string	Имя пользователя

Открыть статью на сайте

Метод Password

Возвращает или задает пароль для использования при аутентификации (getters/setters).

```
get Password(): string
```

```
set Password(passwd: string)
```

Параметр	Тип	Описание
passwd	string	Пароль

Открыть статью на сайте

Метод ClientCertificate

Возвращает или задает клиентский сертификат, для подключения к службе (getters/setters).

```
get ClientCertificate(): pki.Certificate
```

```
set ClientCertificate(clntCert: pki.Certificate)
```

Параметр	Тип	Описание
clntCert	Certificate	Клиентский сертификат

Открыть статью на сайте

Метод ProxyAuthType

Возвращает или задает тип аутентификации для прокси-сервера (getters/setters).

```
get ProxyAuthType(): number
```

```
set ProxyAuthType(authType: number)
```

Параметр	Тип	Описание
authType	number	Тип аутентификации (enumAuthTypes)

Открыть статью на сайте

Метод ProxyAddress

Возвращает или задает адрес прокси-сервера (getters/setters).

```
get ProxyAddress(): string
```

```
set ProxyAddress(addr: string)
```

Параметр	Тип	Описание
addr	string	Адрес прокси

Открыть статью на сайте

Метод ProxyUserName

Возвращает или задает имя пользователя для использования прокси-сервера (getters/setters).

```
get ProxyUserName(): string
```

```
set ProxyUserName(usrName: string)
```

Параметр	Тип	Описание
usrName	string	Имя пользователя прокси-сервера

Открыть статью на сайте

Метод ProxyPassword

Возвращает или задает имя пользователя для использования при аутентификации для прокси-сервера (getters/setters).

```
get ProxyPassword(): string
```

```
set ProxyPassword(passwd: string)
```

Параметр	Тип	Описание
passwd	string	Пароль

Открыть статью на сайте

Описание класса ModuleInfo

Метод	Возвращаемый тип	Описание
get `version`	string	Возвращает версию модуля
set `name`	string	Возвращает имя модуля
get `cadesEnabled`	boolean	Проверка доступности CAdES

Открыть статью на сайте

Описание класса Tools

Метод	Возвращаемый тип	Описание
`stringFromBase64`	string	Декодирует строку из BASE64
`stringToBase64`	string	Кодирует строку в BASE64

Открыть статью на сайте

Описание класса Filter

Класс используется для фильтрации (поиска) объектов типа PkiItem.

Метод	Возвращаемый тип	Описание
`set types`	void	Задает тип объекта PkiItem
`set providers`	void	Задает тип провайдера
`set categorys`	void	Задает категорию (хранилище)
`set hash`	void	Задает отпечаток (SHA-1)
`set subjectName`	void	Задает имя субъекта
`set subjectFriendlyName`	void	Задает имя субъекта (CN)
`set issuerName`	void	Задает имя издателя
`set issuerFriendlyName`	void	Задает имя издателя (CN)
`set serial`	void	Задает серийный номер

Открыть статью на сайте

Описание класса PkiStore

Метод	Возвращаемый тип	Описание
`addProvider`	void	Добавляет провайдер. В данной версии только `ProviderCryptopro`
`find`	IPkiItem[]	Поиск объектов в хранилище
`getItem`	Certificate или CRL	Возвращает объект по его описанию `PkiItem`
get `certs`	CertificateCollection	Возвращает все сертификаты из хранилища
`addCert`	void	Импорт сертификата в хранилище
`addCrl`	void	Импорт CRL в хранилище
`deleteCert`	void	Удаление сертификата из хранилища
`deleteCrl`	void	Удаление CRL из хранилища

Открыть статью на сайте

Метод addProvider

Добавляет провайдер. В данной ревизии только ProviderCryptopro.

```
addProvider(provider: Provider): void
```

Параметр	Тип	Описание
provider	Provider	Провайдер. В данной версии только ProviderCryptopro.

Открыть статью на сайте

Метод find

Поиск объектов в хранилище.

```
find(ifilter?: IFilter): IPkiItem[]
```

Параметр	Тип	Описание
filter?	IFilter	Необязательный параметр. Параметры фильтрации.

Открыть статью на сайте

Метод getItem

Возвращает объект по его описанию IPkiItem.

```
getItem(item: IPkiItem)
```

Параметр	Тип	Описание
item	IPkiItem	Описание объекта

Открыть статью на сайте

Метод certs

Возвращает все сертификаты из хранилища.

```
get certs():CertificateCollection
```

Открыть статью на сайте

Метод addCert

Импорт сертификата в хранилище.

```
addCert(provider: Provider, category: string, cert: pki.Certificate, contName?: string,
provType?: number): void
```

Параметр	Тип	Описание
provider	Provider	Провайдер. В данной версии только ProviderCryptopro.
category	string	Название хранилища
cert	Certificate	Импортируемый сертификат
contName?	string	Необязательный параметр. Имя контейнера.
provType?	number	Необязательный параметр. Тип провайдера. Для ГОСТ 75, 80, 81.

Открыть статью на сайте

Метод addCert

Импорт CRL в хранилище.

```
addCrl(provider: Provider, category: string, crl: pki.CRL): void
```

Параметр	Тип	Описание
provider	Provider	Провайдер. В данной версии только ProviderCryptopro.
category	string	Название хранилища
crl	CRL	Импортируемый СОС (CRL)

Открыть статью на сайте

Метод deleteCert

Удаление сертификата из хранилища.

```
deleteCert(provider: Provider, category: string, cert: Certificate): void
```

Параметр	Тип	Описание
provider	Provider	Провайдер. В данной версии только ProviderCryptopro.
category	string	Название хранилища
cert	Certificate	Удаляемый сертификат

Открыть статью на сайте

Метод deleteCrl

Удаление СОС (CRL) из хранилища.

```
deleteCrl(provider: Provider, category: string, crl: pki.CRL): void
```

Параметр	Тип	Описание
provider	Provider	Провайдер. В данной версии только ProviderCryptopro.
category	string	Название хранилища
crl	CRL	Удаляемый СОС (CRL)

Открыть статью на сайте

Описание класса ProviderCryptopro

Класс при инициализации перечитывает объекты их локального хранилища. Методы класса `ProviderCryptopro` используются из класса `PkiStore`. На уровень `typescript` вынесен один метод.

Метод	Возвращаемый тип	Описание
`hasPrivateKey`	void	Проверка сертификата на наличие привязки с закрытым ключом

Открыть статью на сайте

IPkiKey

Свойство	Тип	Описание
encrypted?	boolean	Зашифрован ли ключ

Открыть статью на сайте

IPkiCrl

Свойство	Тип	Описание
authorityKeyid?	string	Authority Key Identifier
crlNumber?	string	Номер CRL
issuerName?	string	Полное имя издателя
issuerFriendlyName?	string	CN имени издателя
lastUpdate?	string	Дата последнего обновления
nextUpdate?	string	Дата следующего обновления

Открыть статью на сайте

IPkiRequest

Свойство	Тип	Описание
subjectName?	string	Полное имя субъекта
subjectFriendlyName?	string	CN имени субъекта
key?	string	Привязка с закрыты ключом

Открыть статью на сайте

IPkiCertificate

Свойство	Тип	Описание
subjectName?	string	Полное имя субъекта
subjectFriendlyName?	string	CN имени субъекта
issuerName?	string	Полное имя издателя
issuerFriendlyName?	string	CN имени издателя
notAfter?	string	Дата окончания действия сертификата
notBefore?	string	Дата начала действия сертификата
serial?	string	Серийный номер сертификата
key?	string	Привязка с закрыты ключом
organizationName?	string	Организация
signatureAlgorithm?	string	Алгоритм подписи
signatureDigestAlgorithm?	string	Хэш алгоритм подписи
publicKeyAlgorithm?	string	Алгоритм публичного ключа

Открыть статью на сайте

IPkiItem

extends IPkiCrl, IPkiCertificate, IPkiRequest, IPkiKey

Свойство	Тип	Описание
format	string	PEM, DER
type	string	CRL, CERTIFICATE, KEY, REQUEST
uri	string	Ссылка на объект. В данной версии не используется.
provider	string	Имя провайдера
category	string	Категория (название хранилища)
hash	string	Хэш (SHA-1 отпечаток)

Открыть статью на сайте

Описание класса Logger

Метод	Возвращаемый тип	Описание
`start`	void	Старт логгирования
`stop`	void	Завершение логгирования
`clear`	void	Очистка файла журнала (log)

Открыть статью на сайте

Метод start

Старт логгирования операций и ошибок. Запись ведется в файл.

```
start(filename: string, level: LoggerLevel = DEFAULT_LOGGER_LEVEL): void
```

Параметр	Тип	Описание
filename	string	Путь до файла для записи журнала
Level?	LoggerLevel	Необязательный параметр. Уровень логгирования. Значение по умолчанию: LoggerLevel.ALL

Открыть статью на сайте

Руководство программиста (SDK)

Настоящее руководство содержит описание входящего в состав КриптоАРМ ГОСТ модуля, реализующего программный интерфейс **trusted-crypto**.

Интерфейс **trusted-crypto** является встроенным в ПО КриптоАРМ программным интерфейсом языка Javascript, включающим в себя набор объектов с методами работы с сертификатами, подписи и шифрования в формате CMS, установки TLS-соединения.