КриптоАРМ ГОСТ

Руководство программиста

Версия выпуска 1.3.0

ОГЛАВЛЕНИЕ

Общие сведения о программном продукте

КриптоАРМ ГОСТ - это универсальное приложение с графическим пользовательским интерфейсом для выполнения операций по созданию и проверке электронной подписи файлов, шифрования и расшифрования, управления сертификатами, размещенных в хранилищах криптопровайдеров.

Приложение КриптоАРМ ГОСТ является кроссплатформенным, и представлено различными установочными дистрибутивами под платформы: Microsoft Windows, Linux, OSX. На каждой из платформ реализована поддержка российских криптографических стандартов (в том числе ГОСТ Р 34.10-2012) посредством использования криптопровайдера КриптоПро CSP.

В приложении поддерживается работа с ключевыми носителями Рутокен и JaCarta через криптопровайдер КриптоПро CSP.

Структура программного продукта КриптоАРМ ГОСТ 1.3.0

Поддерживаемые операционные системы

Сборка модуля может быть выполнена на одной из перечисленных ниже операционных систем при условии установки необходимого программного окружения для компиляции:

- Microsoft Windows 7 64bit.
- Microsoft Windows 10 64bit/32bit.
- Ubuntu 14.04 64bit/32bit.
- Ubuntu 16.04 64bit/32bit.
- CentOS 7.0 64bit/32bit.
- Rosa Fresh R8 64bit/32bit.
- Rosa Fresh R9 64bit/32bit.
- Rosa Enterprise Desktop (RED) X3 64bit.
- Гослинукс 6.4 64bit.
- Astra Linux Common Edition 64bit.
- Astra Linux Special Edition 64bit.
- Ось 2.1 64bit.
- ALT Linux 7.0 Centaurus 64bit/32bit.
- Mac OS X 10.10, 10.11, 10.12 (64 bit).

Не исключается возможность работы приложения на других платформах, не входящих в представленный выше перечень. Но следует учесть, что для работы с ГОСТ алгоритмами

необходима установка криптопровайдера КриптоПРО CSP на выбранную платформу. Тестирование корректности работы приложения на иных платформах возлагается на самого пользователя. Для этих целей можно обратиться в компанию разработчика приложения и запросить временный лицензионный ключ.

1. Запуск в режиме отладки (с использованием консоли DevTools)

1.1. Отладочный режим на платформе Microsoft Windows

Для запуска командной строки нажать Win+R. Ввести команду cmd и ОК

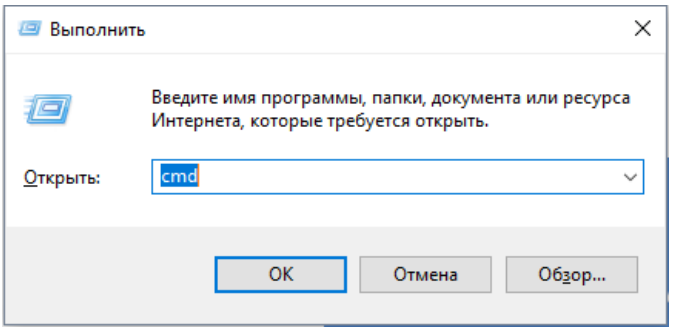

Рис. 1.1.1. Диалог для запуска приложений

В открывшемся окне ввести команду запуска приложения КриптоАРМ ГОСТ (рис. 1.1.1):

"C:\Program Files\CryptoARM GOST\CryptoARM_GOST.exe" devtools logcrypto

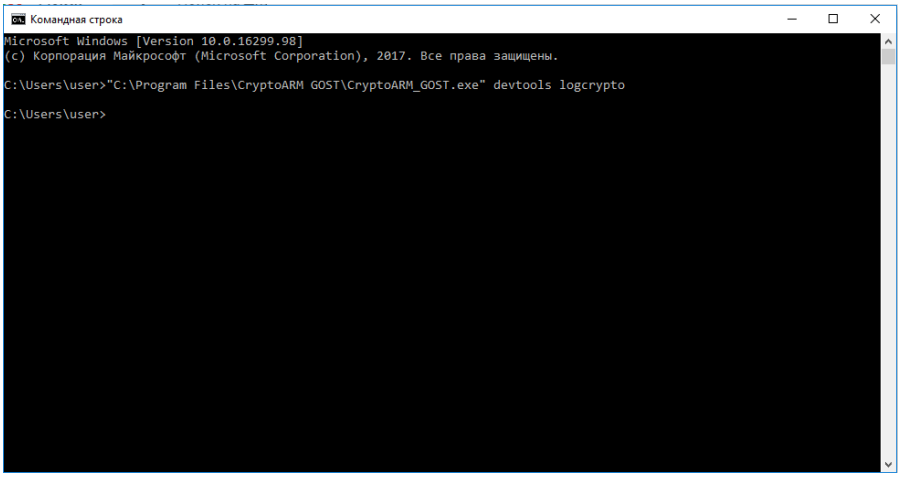

Рис. 1.1.2. Диалог командной строки

В результате выполнения этой команды откроется приложение КриптоАРМ ГОСТ с дополнительной панелью управления, которая представлена на рис. 1.1.3 и сохранением информации об операциях в журнал логирования.

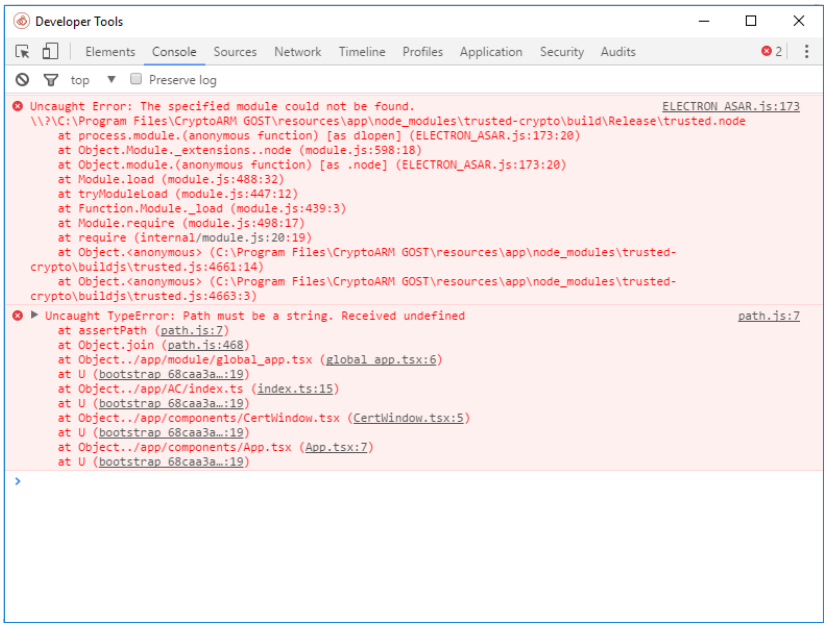

Рис. 1.1.3. Окно с вкладкой консоли управления

При выполнении операции, приводящей к ошибке, в открывшемся дополнительном окне управления на вкладке Console отобразится текст ошибки.

Журнал логирования представляет собой текстовый файл cryptoarm_gost.log, который располагается в каталоге пользователя .Trusted.

1.2. Отладочный режим на платформе OSX

Для получения доступа к дополнительной панели управления приложением и включением режима логирования, в терминале ОС Linux нужно ввести команду (рис. 1.2.1):

/opt/cryptoarm_gost/CryptoARM_GOST devtools logcrypto

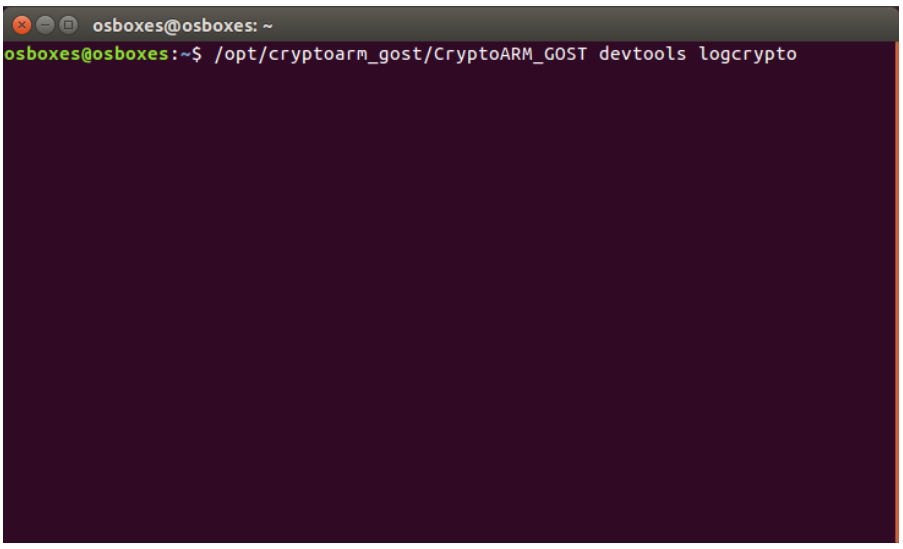

Рис. 1.2.1. Окно терминала

При выполнении операции, приводящей к ошибке, в открывшемся дополнительном окне на вкладке Console отобразится текст ошибки (рис. 1.2.2).

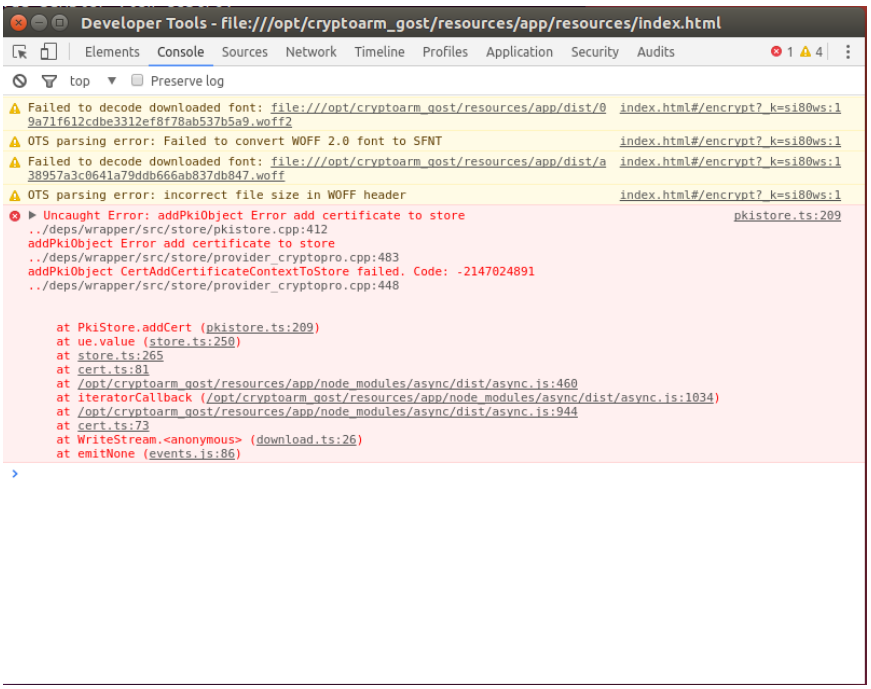

Рис. 1.2.2. Окно с вкладкой консоли управления

При выполнении операции, приводящей к ошибке, в открывшемся дополнительном окне управления на вкладке Console отобразится текст ошибки.

Журнал логирования представляет собой текстовый файл cryptoarm_gost.log, который располагается в каталоге пользователя .Trusted.

1.3. Отладочный режим на платформе Linux

Для получения доступа к дополнительной панели управления приложением и включением режима логирования, в терминале ОС OS X ввести команду (рис. 1.3.1):

/Applications/CryptoARM-GOST.app/Contents/MacOS/CryptoARM_GOST devtools logcrypto

Рис. 1.3.1. Окно терминала

При выполнении операции, приводящей к ошибке, в открывшемся дополнительном окне на вкладке Console отобразится текст ошибки (рис. 1.3.2).

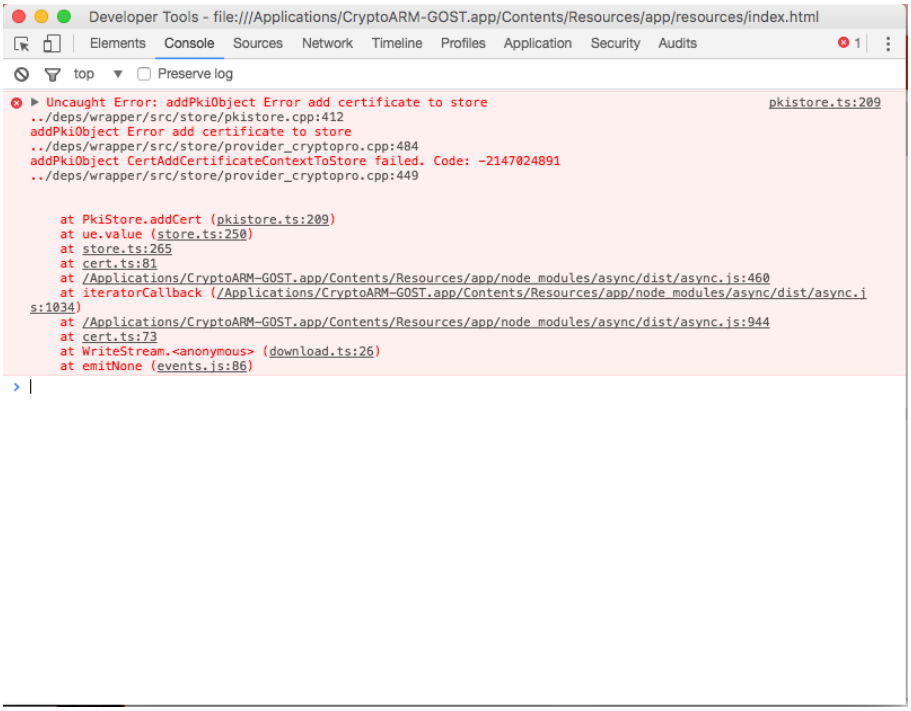

Рис. 1.3.2. Окно с вкладкой консоли управления

При выполнении операции, приводящей к ошибке, в открывшемся дополнительном окне управления на вкладке Console отобразится текст ошибки.

Журнал логирования представляет собой текстовый файл cryptoarm_gost.log, который располагается в каталоге пользователя .Trusted.

2. Описание API встроенного веб-сервера КриптоАРМ ГОСТ

В программном продукте КриптоАРМ ГОСТ имеется встроенный веб-сервер, который запускается сразу при открытии приложения и прослушивает несколько портов localhost. Встроенный сервер позволяет поддержать взаимодействие приложения с любым клиентским браузером через localhost для выполнения операций подписи и шифрования файлов. Таким образом, данное решение может рассматриваться как клиентская криптографическая утилита, вызываемая запросом из кода JavaScript.

Примеры формирования таких запросов представлены в директории demo репозитория https://github.com/TrustedPlus/esign/tree/cryptoarm_gost .

Наиболее значимые сценарии работы такого решения представлены в виде API (все запросы API реализованы в формате JSON RPC 2.0[1]).

2.1. Запросы подписания и проверки подписи файлов

Запрос Sign

Обработка запроса на подпись файлов выполняется по следующему сценарию. С клиента на сервер КриптоАРМ ГОСТ передается запрос sign с вложенным массивом ссылок на файлы. Сервер разбирает этот запрос и инициирует запуск визуальной части приложения (мастера подписи\проверки подписи файлов КриптоАРМ ГОСТ), в котором отображаются требуемые для подписи файлы общим списком.

Файлы из данного списка скачиваются по тем прямым ссылкам, которые переданы в теле запроса и помещаются во временную директорию. Далее пользователь просматривает файлы (или их содержимое) и подтверждает операцию их подписания. В этом случае файлы подписываются и передаются запросом на сторонний контроллер загрузки[2], которым также указан в теле запроса.

Возможен также другой вариант – когда пользователь отказывается (отменяет) от подписи файлов. Тогда на сторону клиента пересылается сообщение, содержащее соответствующий код ошибки.

Формат запроса:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
method	*"sign"*	Указание метода для выполнения обработки
params	*Object[1]*	Вложенный объект с данными, см. расшифровку ниже.
extra	*Object[2]*	Вложенный объект с данными, см. расшифровку ниже
controller	*string*	URL-адрес контроллера для отправки результатов подписи файлов.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object[1]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
token	*string*	Токен аутентификации для гарантированного доступа к файлам

[1] http://www.jsonrpc.org/specification

[2] Предполагается, что информационная система, которая задействует КриптоАРМ ГОСТ в качестве средства подписи имеет помимо клиента серверную часть с хранилище подписанных файлов, либо их обработчик.

files	*Object[3]*	Вложенный объект с данными, см. расшифровку ниже

Спецификация *Object[2]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
role	*"CLIENT"*	

Спецификация *Object[3]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
name	*"string"*	Оригинальное наименование файла, которое должно быть представлено пользователю
url	*"string"*	URL-адрес прямой ссылки на скачивание файла

Например,

{"json-rpc": "2.0", "method": "sign", "params": {"token": "string", "files": {

"0": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=1", "id": 1 },

"1": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=2", "id": 2 },

"2": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=3", "id": 3 }},

"extra": { "role": "CLIENT" }, "controller": "https://bitrix.ru/ajax.php" }, "id": "11"}

Формат ответа, содержащего сообщение об ошибке:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
error	*Object[1]*	Вложенный объект с данными, см. расшифровку ниже.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object[1]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
code	*number*	Числовой код ошибки в результате обработки запроса
message	*string*	Текстовая расшифровка ошибки

Например,

{"jsonrpc": "2.0", "error": {"code": -32700, "message": "Ошибка разбора запроса"}, "id": "11"}

Формат ответа, содержащего сообщение об успешном выполнении операции:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
result	*Object[1]*	Вложенный объект с данными, см. расшифровку ниже.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object[1]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
code	*number*	Числовой код результата выполнения операции
message	*string*	Текстовая расшифровка сообщения

Например,

{"jsonrpc": "2.0", "result": {"code": 200, "message": "Операция подписи выполнена"}, "id": "11"}

Запрос Verify

Обработка запроса на проверку подписи файлов выполняется по следующему сценарию. С клиента на сервер КриптоАРМ ГОСТ передается запрос verify с вложенным массивом ссылок на файлы. Сервер разбирает этот запрос и инициирует запуск визуальной части приложения (мастера подписи\проверки подписи файлов КриптоАРМ ГОСТ), в котором отображаются требуемые для проверки подписи файлы общим списком.

Файлы из данного списка скачиваются по тем прямым ссылкам, которые переданы в теле запроса и помещаются во временную директорию. Далее пользователь выполняет проверку подписи файлов. В этом случае у файлов проверяется подпись и на клиентскую сторону передается сообщение.

Возможен также другой вариант – когда пользователь отказывается от проверки подписи файлов. Тогда на сторону клиента пересылается сообщение, содержащее соответствующий код ошибки.

Формат запроса:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
method	*"verify"*	Указание метода для выполнения обработки
params	*Object[1]*	Вложенный объект с данными, см. расшифровку ниже.
extra	*Object[2]*	Вложенный объект с данными, см. расшифровку ниже
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object[1]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
token	*string*	Токен аутентификации для гарантированного доступа к файлам
files	*Object[3]*	Вложенный объект с данными, см. расшифровку ниже

Спецификация *Object[2]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
role	*"CLIENT"*	

Спецификация *Object[3]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
name	*"string"*	Оригинальное наименование файла, которое должно быть представлено пользователю
url	*"string"*	URL-адрес прямой ссылки на скачивание файла

Например,

{"json-rpc": "2.0", "method": "verify", "params": {"token": "string", "files": {

"0": { "name": "filename.txt.sig", "url": "https://bitrix.ru/ajax.php?command=content&id=1", "id": 1 },

"1": { "name": "filename.txt.sig", "url": "https://bitrix.ru/ajax.php?command=content&id=2", "id": 2 },

"2": { "name": "filename.txt.sig", "url": "https://bitrix.ru/ajax.php?command=content&id=3", "id": 3 }},

"extra": { "role": "CLIENT" }}, "id": "12"}

Формат ответа, содержащего сообщение об ошибке:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
error	*Object*[1]	Вложенный объект с данными, см. расшифровку ниже.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object*[1]

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
code	*number*	Числовой код ошибки в результате обработки запроса
message	*string*	Текстовая расшифровка ошибки

Например,

{"jsonrpc": "2.0", "error": {"code": -32700, "message": "Ошибка разбора запроса"}, "id": "12"}

Формат ответа, содержащего сообщение об успешном выполнении операции:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
result	*Object*[1]	Вложенный объект с данными, см. расшифровку ниже.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object*[1]

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
code	*number*	Числовой код результата выполнения операции
message	*string*	Текстовая расшифровка сообщения

Например,

{"jsonrpc": "2.0", "result": {"code": 201, "message": "Операция проверки подписи выполнена"}, "id": "12"}

2.2. ЗАПРОСЫ ШИФРОВАНИЯ И РАСШИФРОВАНИЯ ФАЙЛОВ

Запрос Encrypt

Обработка запроса на шифрование файлов выполняется по следующему сценарию. С клиента на сервер КриптоАРМ ГОСТ передается запрос encrypt с вложенным массивом ссылок на файлы. Сервер разбирает этот запрос и инициирует запуск визуальной части приложения (мастера подписи файлов КриптоАРМ ГОСТ), в котором отображаются требуемые для шифрования файлы общим списком.

Файлы из данного списка скачиваются по тем прямым ссылкам, которые переданы в теле запроса и помещаются во временную директорию. Далее пользователь просматривает файлы (или их содержимое) и подтверждает операцию их шифрования в адрес выбранных получателей. В этом случае файлы подписываются и передаются запросом на сторонний контроллер загрузки[3], которым также указан в теле запроса.

Возможен также другой вариант – когда пользователь отказывается (отменяет) от

[3] Предполагается, что информационная система, которая задействует КриптоАРМ ГОСТ в качестве средства подписи имеет помимо клиента серверную часть с хранилище подписанных файлов, либо их обработчик.

шифрования переданных файлов. Тогда на сторону клиента пересылается сообщение, содержащее соответствующий код ошибки.

Формат запроса:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
method	*"encrypt"*	Указание метода для выполнения обработки
params	*Object[1]*	Вложенный объект с данными, см. расшифровку ниже.
extra	*Object[2]*	Вложенный объект с данными, см. расшифровку ниже
controller	*string*	URL-адрес контроллера для отправки результатов подписи файлов.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object[1]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
token	*string*	Токен аутентификации для гарантированного доступа к файлам
files	*Object[3]*	Вложенный объект с данными, см. расшифровку ниже

Спецификация *Object[2]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
role	*"CLIENT"*	

Спецификация *Object[3]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
name	*"string"*	Оригинальное наименование файла, которое должно быть представлено пользователю
url	*"string"*	URL-адрес прямой ссылки на скачивание файла

Например,

{"json-rpc": "2.0", "method": "encrypt", "params": {"token": "string", "files": {

"0": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=1", "id": 1 },

"1": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=1", "id": 2 },

"2": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=1", "id": 3 }},

"extra": { "role": "CLIENT" }, "controller": "https://bitrix.ru/ajax.php" }, "id": "13"}

Формат ответа, содержащего сообщение об ошибке:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
error	*Object[1]*	Вложенный объект с данными, см. расшифровку ниже.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object[1]*

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
code	*number*	Числовой код ошибки в результате обработки запроса
message	*string*	Текстовая расшифровка ошибки

Например,

{"jsonrpc": "2.0", "error": {"code": -32700, "message": "Ошибка разбора запроса"}, "id": "13"}

Формат ответа, содержащего сообщение об успешном выполнении операции:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
result	*Object*[1]	Вложенный объект с данными, см. расшифровку ниже.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object*[1]

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
code	*number*	Числовой код результата выполнения операции
message	*string*	Текстовая расшифровка сообщения

Например,

{"jsonrpc": "2.0", "result": {"code": 200, "message": "Операция подписи выполнена"}, "id": "13"}

Запрос Decrypt

Обработка запроса на расшифрование файлов выполняется по следующему сценарию. С клиента на сервер КриптоАРМ ГОСТ передается запрос decrypt с вложенным массивом ссылок на файлы. Сервер разбирает этот запрос и инициирует запуск визуальной части приложения (мастера подписи файлов КриптоАРМ ГОСТ), в котором отображаются требуемые для подписи файлы общим списком.

Файлы из данного списка скачиваются по тем прямым ссылкам, которые переданы в теле запроса и помещаются во временную директорию. Далее пользователь просматривает файлы (или их содержимое) и подтверждает операцию их расшифрования. В этом случае файлы подписываются и передаются запросом на сторонний контроллер загрузки[4], которым также указан в теле запроса.

Возможен также другой вариант – когда пользователь отказывается (отменяет) от расшифрования файлов. Тогда на сторону клиента пересылается сообщение, содержащее соответствующий код ошибки.

Формат запроса:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
method	*"decrypt"*	Указание метода для выполнения обработки
params	*Object*[1]	Вложенный объект с данными, см. расшифровку ниже.
extra	*Object*[2]	Вложенный объект с данными, см. расшифровку ниже
controller	*string*	URL-адрес контроллера для отправки результатов подписи файлов.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object*[1]

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
token	*string*	Токен аутентификации для гарантированного доступа к файлам
files	*Object*[3]	Вложенный объект с данными, см. расшифровку ниже

Спецификация *Object*[2]

[4] Предполагается, что информационная система, которая задействует КриптоАРМ ГОСТ в качестве средства подписи имеет помимо клиента серверную часть с хранилище подписанных файлов, либо их обработчик.

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
role	*"CLIENT"*	

Спецификация *Object*[3]

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
name	*"string"*	Оригинальное наименование файла, которое должно быть представлено пользователю
url	*"string"*	URL-адрес прямой ссылки на скачивание файла

Например,

{"json-rpc": "2.0", "method": "decrypt", "params": {"token": "string", "files": {

"0": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=1", "id": 1 },

"1": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=1", "id": 2 },

"2": { "name": "filename.txt", "url": "https://bitrix.ru/ajax.php?command=content&id=1", "id": 3 }},

"extra": { "role": "CLIENT" }, "controller": "https://bitrix.ru/ajax.php" }, "id": "14"}

Формат ответа, содержащего сообщение об ошибке:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
error	*Object*[1]	Вложенный объект с данными, см. расшифровку ниже.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object*[1]

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
code	*number*	Числовой код ошибки в результате обработки запроса
message	*string*	Текстовая расшифровка ошибки

Например,

{"jsonrpc": "2.0", "error": {"code": -32700, "message": "Ошибка разбора запроса"}, "id": "14"}

Формат ответа, содержащего сообщение об успешном выполнении операции:

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
json-rpc	*"2.0"*	Обязательное значение по спецификации
result	*Object*[1]	Вложенный объект с данными, см. расшифровку ниже.
id	*number*	Уникальный идентификатор запроса в очереди

Спецификация *Object*[1]

КЛЮЧ	ЗНАЧЕНИЕ	ОПИСАНИЕ
code	*number*	Числовой код результата выполнения операции
message	*string*	Текстовая расшифровка сообщения

Например,

{"jsonrpc": "2.0", "result": {"code": 200, "message": "Операция подписи выполнена"}, "id": "14"}

3. Описание API нативного модуля TrustedCrypto в составе продукта КриптоАРМ ГОСТ

Все интерфейсные функции являются методами классов и объединяются в несколько больших групп по своему назначению. Данные группы рассматриваются как пакеты.

В модуле представлены пакеты:

- CMS – предоставляет набор методов для выполнения операций с цифровой подписью в формате CMS[2].
- PKI – предоставляет набор методов для работы с криптографическими объектами, такими как цифровые сертификаты, списки отзыва, идентификаторы, ключевые контейнеры и т.п.
- PKISTORE – предоставляет набор методов для работы с криптографическими провайдерами, посредством предоставляемого API.
- COMMON – набор методов для управления контекстом библиотеки OpenSSL.
- UTILS – набор вспомогательных методов для работы с лицензией и логированием операций.

Далее представлено подробное описание интерфейсных функций модуля в рамках обозначенных пакетов.

3.1. ПАКЕТ CMS

Пакет предоставляет необходимые методы для создания подписи в формате CMS. Стандарт CMS описывает структуру криптографических сообщений, включающих в себя защищенные данные вместе со сведениями, необходимыми для их корректного открытия или использования. Например, в сообщении размещаются защищенные данные, информация об алгоритме хеширования и подписи, времени подписи, сертификате открытого ключа, цепочке сертификации и т.д. Некоторые из указанных атрибутов носят опциональный характер, но приложение может само определить необходимость их наличия.

Подпись, описанная стандартом CMS, характеризуется следующими особенностями:

- Данные могут быть подписаны несколькими сторонами (множественная подпись). В таком случае в сообщении будут присутствовать несколько структур SignerInfo с информацией о подписывающих сторонах: значением подписи и необходимой для проверки ее подлинности информацией.
- Тип данных никак не регламентируется, лишь уточняется, что в качестве данных может быть сообщение формата CMS, то есть подписанное одной стороной сообщение может быть целиком подписано другой стороной.
- Подписывать можно не только данные, но и некоторые атрибуты сообщения – хеш сообщения (digest message), время подписи (signing time), значение другой подписи (countersignature).
- Открытый ключ подписывающей стороны может быть несертифицированным.
- Подпись может отсутствовать и вовсе.

[2] RFC 5652

Класс SignedData

В классе реализованы следующие интерфейсы, свойства и методы:

Интерфейсы

НАЗВАНИЕ	ПАРАМЕТРЫ	ОПИСАНИЕ
ISIGNEDDATACONTENT	type: SignedDataContentType data: string | Buffer	

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
CONTENT	ISignedDataContent	Возвращает или задает содержимое подписи
POLICIES	Array<string>	Возвращает или задает политики подписи

Методы

МЕТОД	ОПИСАНИЕ
ISDETACHED()	Возвращает true если подпись открепленная
CERTIFICATES()	Возвращает коллекцию сертификатов
SIGNERS()	Возвращает коллекцию подписчиков
LOAD(STRING, DATAFORMAT)	Выполняет чтение подписи из файла
IMPORT(BUFFER, DATAFORMAT)	Выполняет чтение подписи из памяти
SAVE(STRING, DATAFORMAT)	Выполняет сохранение подписи в файл
EXPORT(DATAFORMAT)	Выполняет сохранение подписи в память
CREATESIGNER(CERTIFICATE, KEY)	Создает нового подписчика
VERIFY(CERTIFICATECOLLECTION)	Выполняет проверку подписи
SIGN()	Выполняет подпись данных

Пояснения по использованию свойств и методов класса.

isDetached(): boolean

Метод возвращает значение true если подпись открепленная. Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("test.sig", trusted.DataFormat.PEM);
cms.isDetached();
```

certificates(index: number): Certificate

Метод возвращает сертификат по индексу. В методе используется параметр

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
var  certificate = cms.certificates(0);
console.log(certificate.signatureAlgorithm); // sha256
```

certificates(): CertificateCollection

Метод возвращает коллекцию сертификатов CertificateCollection. Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
var certificates = cms.certificates();
console.log(certificates.length); // 1
```

signers(index: number): Signer

Метод возвращает подписчика по индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("sigdoc.sig", trusted.DataFormat.PEM);
var signer = cms.signers(0);
console.log("Signer digest name:", signer.digestAlgorithm.name); // Signer digest name: sha1
```

signers(): SignerCollection

Метод возвращает коллекцию подписчиков. Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("sigdoc.sig", trusted.DataFormat.PEM);
var signers = cms.signers();
for (var i = 0; i < signers.length; i++){ var signer = signers.items(i);
        console.log("Signer digest name:", signer.digestAlgorithm.name); // Signer digest name: sha1
}
```

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

Метод выполняет чтение подписи из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER

Например,

```
var trusted = require("trusted-crypto");

var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
```

static load(filename: string, format: DataFormat): SignedData

Метод выполняет чтение подписанных данных из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER

import(buffer: Buffer, format: DataFormat): void

Метод выполняет чтение подписи из памяти. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
BUFFER	buffer	Буфер
FORMAT	DataFormat	Формат данных. По умолчанию DER

static import(buffer: Buffer, format: DataFormat): SignedData

Метод выполняет чтение подписанных данных из памяти. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
BUFFER	buffer	Буфер
FORMAT	DataFormat	Формат данных. По умолчанию DER

export(format: DataFormat): Buffer

Метод выполняет сохранение подписи в память. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FORMAT	DataFormat	Формат данных. По умолчанию DER

Например,

```
var trusted = require("trusted-crypto");

var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
```

```
var buf = cms.export(trusted.DataFormat.PEM);
```

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

Метод выполняет сохранение подписи в файл. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных. По умолчанию DER

Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
cms.save("testsig1.sig", trusted.DataFormat.PEM);
```

createSigner(cert: Certificate, key: Key): Signer

Метод создает нового подписчика. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат подписчика
KEY	Key	Закрытый ключ сертификата подписчика

Например,

```
var trusted = require("trusted-crypto");
var cert = trusted.pki.Certificate.load("cert1.crt", trusted.DataFormat.PEM);
var key = trusted.pki.Key.readPrivateKey("cert1.key", trusted.DataFormat.PEM, "");
var sd = new trusted.cms.SignedData();
var signer = sd.createSigner(cert, key);
```

verify(certs: CertificateCollection): boolean

Метод выполняет проверку подписи. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERTS	CertificateCollection	Коллекция дополнительных сертификатов

Метод может быть вызван без параметра.

Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
cms.verify();
```

sign(): void

Метод выполняет создание подписи. Например,

```
var trusted = require("trusted-crypto");
var cert = trusted.pki.Certificate.load("./test/cert.crt", trusted.DataFormat.PEM);
var key = trusted.pki.Key.readPrivateKey("./test/cert.key", trusted.DataFormat.PEM, "");
var sd = new trusted.cms.SignedData();
var signer = sd.createSigner(cert, key); sd.content = {data: 'Hellow word'};
sd.sign();
sd.save("testsig.sig", trusted.DataFormat.PEM);
```

Класс Signer

В классе реализованы следующие интерфейсы, свойства и методы:

Интерфейсы

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
CERTIFICATE	Certificate	Задает или возвращает сертификат подписчика
DIGESTALGORITHM	Algorithm	Возвращает хэш алгоритм проверки содержимого
SIGNERID	SignerId	Возвращает идентификатор подписчика

Методы

МЕТОД	ОПИСАНИЕ
SIGNEDATTRIBUTES(SIGNERATTRIBUTECOLLECTION)	Возвращает коллекцию подписанных атрибутов
UNSIGNEDATTRIBUTES(SIGNERATTRIBUTECOLLECTION)	Возвращает коллекцию неподписанных атрибутов
VERIFYCONTENT()	Возвращает результат проверки подписанного контента
VERIFY()	Возвращает результат проверки атрибутов подписи

Пояснения по использованию свойств и методов класса.

signedAttributes(): SignerAttributeCollection

Метод возвращает коллекцию подписанных атрибутов. Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
var signer = cms.signers(0);
var signedAttributes = signer.signedAttributes();
console.log(signer.digestAlgorithm.name);// sha1 console.log(signer.signedAttributes.length);//1
```

signedAttributes(index: number): Attribute

Метод возвращает атрибут из коллекции по заданному индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

unsignedAttributes(): SignerAttributeCollection

Метод возвращает коллекцию неподписанных атрибутов.

unsignedAttributes(index: number): Attribute

Метод возвращает атрибут из коллекции по заданному индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ

INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
var signer = cms.signers(0);
var unsignedAttributes = signer.unsignedAttributes();
console.log(unsignedAttributes.length);//-1
```

verifyContent(v: ISignedDataContent): boolean

Метод возвращает true, если подписанный контент не был изменен. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
V	any	Контент

verify(): boolean

Метод возвращает результат проверки атрибутов подписи.

КЛАСС SIGNERCOLLECTION

В классе реализованы следующие интерфейсы, свойства и методы:

ИНТЕРФЕЙСЫ

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
LENGTH	number	Возвращает размер коллекции

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
ITEMS	Возвращает элемент из коллекции по заданному индексу

Пояснения по использованию свойств и методов класса.

items(index: number): Signer

Метод возвращает элемент из коллекции по заданному индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require("trusted-crypto");
var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);
var signers = cms.signers();
var signer = signers.items(0); console.log(signers.length);// 1
console.log(signer.digestAlgorithm.name); //sha1
```

КЛАСС SIGNERATRIBUTECOLLECTION

Класс предназначен для создания представления коллекции атрибутов подписчика. В классе реализованы следующие свойства и методы:

СВОЙСТВА

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
LENGTH	number	Возвращает размер коллекции

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
ITEMS	Возвращает элемент из коллекции по заданному индексу
PUSH	Добавляет новый элемент в коллекцию
REMOVEAT	Удаляет элемент из коллекции по заданному индексу

Пояснения по использованию свойств и методов класса.

push(attr: Attribute): void

Метод добавляет новый элемент в коллекцию. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
ATTR	Attribute	Новый элемент коллекции

removeAt(index: number): void

Метод удаляет элемент из коллекции по заданному индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

items(index: number): Attribute

Метод возвращает элемент коллекции по заданному индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require("trusted-crypto");

var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);

var signer = cms.signers(0);

var signedAttributes = signer.signedAttributes();

console.log(signedAttributes.length);//-1
```

КЛАСС SIGNERID

Класс предназначен для представления идентификационной информации подписчика. В классе реализованы следующие свойства:

СВОЙСТВА

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
ISSUERNAME()	string	Возвращает полное имя эмитента
SERIALNUMBER()	string	Возвращает серийный номер
KEYID()	string	Возвращает идентификатор ключа

Например,

```
var trusted = require("trusted-crypto");

var cms = trusted.cms.SignedData.load("testsig.sig", trusted.DataFormat.PEM);

var signer = cms.signers(0);

var signerId = signer.signerId; console.log(typeof signerId.issuerName); //string

console.log(typeof signerId.serialNumber); // string

console.log(typeof signerId.keyId); //string
```

КЛАСС CMSRECIPIENTINFO

В классе реализованы следующие свойства и методы:

СВОЙСТВА

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
ISSUERNAME	**STRING**	Возвращает полное имя издателя сертификата
SERIALNUMBER	**STRING**	Возвращает серийный номер сертификата

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
KTRICERTCMP	Возвращает результат сравнения сертификатов по CMS_RecipientInfo структуре

Пояснения по использованию свойств и методов класса.

ktriCertCmp(cert: pki.Certificate): number

Метод возвращает результат сравнения сертификатов по CMS_RecipientInfo структуре.

Например,

```
var trusted = require("trusted-crypto");
var cipher = new trusted.pki.Cipher();
var ris = cipher.getRecipientInfos("/encAssym.enc", trusted.DataFormat.PEM);
console.log(ris.length);
ri = ris.items(0);
console.log(ri.issuerName); console.log(ri.serialNumber);
console.log(ri.ktriCertCmp(trusted.pki.Certificate.load("/cert1.crt", trusted.DataFormat.PEM));
```

КЛАСС CMSRECIPIENTINFOCOLLECTION

В классе реализованы следующие свойства и методы:

СВОЙСТВА

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
LENGTH	**NUMBER**	Возвращает размер коллекции получателей

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
PUSH	Добавляет новый элемент в коллекцию
POP	Удаляет последний элемент из коллекции
REMOVEAT	Удаляет элемент по индексу из коллекции

Пояснения по использованию свойств и методов класса.

push(ri: CmsRecipientInfo): void

Метод добавляет новый элемент в коллекцию.

pop(): void

Метод удаляет последний элемент из коллекции.

removeAt(index: number): void

Метод удаляет элемент из коллекции по индексу.

Например:

```
var trusted = require("trusted-crypto"); var cipher = new trusted.pki.Cipher();
var ris = cipher.getRecipientInfos("/encAssym.enc", trusted.DataFormat.PEM);
var r1 = ris.items(0);
ris.push(r1);
ris.push(r1);
ris.pop();
ris.removeAt(1);
```

3.2. ПАКЕТ PKI

КЛАСС ALGORITHM

Класс реализует представление X509_ALGOR. В классе доступны следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.ALGORITHM)	Конструктор с указанием алгоритма – через экземпляр объекта
CONSTRUCTOR(NAME: STRING)	Конструктор с указанием строкового названия алгоритма

СВОЙСТВА

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
NAME	string	Возвращает название алгоритма
TYPEID	Oid	Возвращает идентификатор алгоритма

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
DUPLICATE	Возвращает копию алгоритма
ISDIGEST	Возвращает true если алгоритм предназначен для вычисления хэш

Пояснения по использованию конструкторов, свойств и методов класса.

constructor()

Конструктор для создания экземпляра объекта по умолчанию.

constructor(handle: native.PKI.Algorithm)

Конструктор для создания экземпляра объекта на основе существующего экземпляра (конструктор копированием)

constructor(name: string)

Конструктор для создания экземпляра объекта по указанному имени алгоритма.

duplicate(): Algorithm

Метод возвращает копию алгоритма.

isDigest(): boolean

Метод возвращает true если алгоритм предназначен для вычисления хэша. Например,

var trusted = require("trusted-crypto");

```
var alg = new trusted.pki.Algorithm('SHA');
console.log(alg.typeId.shortName); // SHA

console.log(alg.name);// sha
console.log(alg.duplicate().name); // sha
console.log(alg.isDigest()); //true
```

Класс Attribute

Класс реализует представление X509_ATTR. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR(HANDLE: NATIVE.PKI.ATTRIBUTE)	Конструктор с указанием атрибута – через экземпляр объекта

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
ASNTYPE	number	Задает ASN1 тип атрибута
TYPEID	Oid	Задает идентификатор атрибута

Методы

МЕТОД	ОПИСАНИЕ
DUPLICATE	Возвращает копию алгоритма
EXPORT	Возвращает атрибут в DER кодировке
VALUES	Возвращает коллекцию значений атрибута. Значения представляются в DER формате

Пояснения по использованию конструкторов, свойств и методов класса.

dupicate(): Attribute

Метод возвращает копию атрибута.

export(): Buffer

Метод возвращает атрибут в DER кодировке.

values(): AttributeValueCollection

Метод возвращает коллекцию значений атрибута. Значения представляются в DER формате.

values(index: number): Buffer

Метод возвращает коллекцию значений атрибута. Значения представляются в DER формате.

В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Класс AttributeValueCollection

Класс реализует представление коллекции «X509_ATTR». В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR(HANDLE: NATIVE.PKI.ATTRIBUTEVALUECOLLECTION)	Конструктор с указанием коллекции атрибутов – через экземпляр объекта

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
LENGTH	number	Возвращает количество элементов в коллекции

Методы

МЕТОД	ОПИСАНИЕ
PUSH	Добавляет новый элемент в коллекцию
POP	Удаляет последний элемент из коллекции
REMOVEAT	Удаляет элемент коллекции по заданному индексу
ITEMS	Возвращает элемент из коллекции по заданному индексу

Пояснения по использованию конструкторов, свойств и методов класса.

push(val: Buffer): void

Метод добавляет новый элемент в коллекцию. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
VAL	Buffer	Новое значение коллекции

pop(): void

Метод удаляет последний элемент из коллекции.

removeAt(index: number): void

Метод удаляет элемент коллекции по заданному индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

items(index: number): Buffer

Метод возвращает элемент из коллекции по заданному индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Класс Certificate

Класс реализует представление «X509» сертификата. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.CERTIFICATE)	Конструктор с указанием сертификата – через экземпляр объекта

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
VERSION	number	Возвращает серийный номер сертификата
SERIALNUMBER	string	Возвращает серийный номер сертификата
TYPE	number	Возвращает тип сертификата
KEYUSAGE	number	Возвращает набор флагов KeyUsageFlags, задающих назначение ключа сертификата
ISSUERFRIENDLYNAME	string	Возвращает пользовательское имя издателя сертификата
ISSUERNAME	string	Возвращает полное имя издателя сертификата
SUBJECTFRIENDLYNAME	string	Возвращает пользовательское имя владельца сертификата
SUBJECTNAME	string	Возвращает полное имя владельца сертификата
NOTBEFORE	date	Возвращает время, с которого сертификат считается действительным
NOTAFTER	date	Возвращает время, до которого сертификат считается действительным
THUMBPRINT	string	Возвращает алгоритм подписи
SIGNATUREALGORITHM	string	Возвращает алгоритм подписи
SIGNATUREDIGEST	string	Возвращает хеш алгоритм
ORGANIZATIONNAME	string	Возвращает название организации
OCSPURLS	string[]	Возвращает массив адресов OCSP служб
CAISSUERSURLS	string[]	Возвращает массив адресов CA сертификатов
ISCA	boolean	Возвращает true, если сертивикат CA (испильуется для подписи других сертификатов)
ISSELFSIGNED	boolean	Возвращает true, если сертификат самоподписанный

Методы

МЕТОД	ОПИСАНИЕ
COMPARE	Сравнение сертификатов
EQUALS	Сравнение сертификатов
HASH	Вычисление значения хэша сертификата
DUPLICATE	Создает копию сертификата
LOAD	Чтение сертификата из файла
IMPORT	Чтение сертификата из памяти
EXPORT	Сохранение сертификата в память
SAVE	Сохранение сертификата в файл
DOWNLOAD	Загрузка сертификатов по указанному списку адресов CA сертфикатов

Пояснения по использованию конструкторов, свойств и методов класса.

compare(cert: Certificate): number

Метод выполняет сравнение сертификатов. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат для сравнения

Например,

```
var trusted = require("trusted-crypto");
var cert1 = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM);
 var cert2 = trusted.pki.Certificate.load("test-ru.crt", trusted.DataFormat.PEM);
console.log (cert1.compare(cert2)); // 1 console.log(cert2.compare(cert1)); // -1
console.log(cert1.compare(cert1)); // 0
```

equals(cert: Certificate): boolean

Метод выполняет сравнение сертификатов. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат для сравнения

Например,

```
var trusted = require("trusted-crypto");
var cert1 = trusted.pki.Certificate.load("test.crt"); var cert2 = trusted.pki.Certificate.load("test-ru.crt");
console.log(cert1.equals(cert2)); //false console.log(cert1.equals(cert1)); // true
```

hash(algorithm?: string): String

Метод выполняет вычисление значения хэша сертификата. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
ALGORITHM	string	Имя хэш алгоритма. Опционально. По умолчанию sha1

Например,

```
var trusted = require("trusted-crypto");
var cert1 = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM
console.log(cert1.hash());
```

duplicate(): Certificate

Метод создает копию сертификата. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
ALGORITHM	string	Имя хэш алгоритма. Опционально. По умолчанию sha1

Например,

```
var trusted = require("trusted-crypto");
var cert1 = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM);
var cert2 = cert1.duplicate();
console.log(cert1.thumbprint === cert2.thumbprint); // true
```

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

Метод выполняет чтение сертификата из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER

Например,

```
var trusted = require("trusted-crypto");
var cert = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM);
console.log(cert.serialNumber);
```

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

Метод выполняет чтение сертификата из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER

import(buffer: Buffer, format?: DataFormat): void

Метод выполняет чтение сертификата из памяти. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
BUFFER	Buffer	Буфер памяти
FORMAT	DataFormat	Формат данных.

static import(buffer: Buffer, format?: DataFormat): Certificate

Метод выполняет чтение сертификата из памяти. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
BUFFER	Buffer	Буфер памяти
FORMAT	DataFormat	Формат данных.

export(format?: DataFormat): Buffer

Метод выполняет сохранение сертификата в память. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FORMAT	DataFormat	Формат данных.

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

Метод выполняет сохранение сертификата в файл. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных.

Например,

```
var trusted = require("trusted-crypto");

var cert = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM);

cert.save ('test_new.crt', trusted.DataFormat.PEM);
```

download(urls: string[],pathForSave: string, done: (err: Error, certificate: Certificate)): void

Метод выполняет загрузку сертификатов. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
URLS	string[]	Массив адресов CA сертификатов
PATHFORSAVE	string	Путь к для сохранения загружаемого файла
DONE	function	callback

Например,

```
var trusted = require("trusted-crypto");

var cert = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM);

var url = cert.CAIssuersUrls;

var path_to_file = /tmp/;
```

trusted.pki.Certificate.download(url, path_to_file, function(err, cert));

КЛАСС CERTIFICATIONREQUEST

Класс предназначен для создания ASN.1 структуры CertificationRequest, которая представлена как:

CertificationRequest ::= SEQUENCE { certificationRequestInfo CertificationRequestInfo, signatureAlgorithm AlgorithmIdentifier{{ SignatureAlgorithms }}, signature BIT STRING }

В классе реализованы следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.CERTIFICATIONREQUEST)	Конструктор копированием экземпляра объекта CertificationRequest

СВОЙСТВА

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
PEMSTRING	Buffer	Получение запроса на сертификат

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
LOAD	Чтение запроса из файла
SIGN	Подпись запроса
VERIFY	Проверка запроса на сертификат

Пояснения по использованию конструкторов, свойств и методов класса.

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

Метод выполняет чтение запроса из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER

Например,

```
var trusted = require("trusted-crypto");

var cr = new trusted.pki.CertificationRequest();

cr.load("test.csr", trusted.DataFormat.PEM);

console.log(cr.PEMString);
```

static load(filename: string, format?: DataFormat): CertificationRequest

Метод выполняет чтение запроса из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ

FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER

sign(key: Key): void

Метод выполняет подпись запроса. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
KEY	Key	Закрытый ключ

verify(): boolean

Метод выполняет проверку подписи запроса на сертификат.

КЛАСС CERTIFICATIONREQUESTINFO

Класс для создания ASN.1 структуры CertificationRequestInfo, которая представлена как:

CertificationRequestInfo ::= SEQUENCE { version INTEGER { v1(0) } (v1,...), subject Name, subjectPKInfo SubjectPublicKeyInfo{{ PKInfoAlgorithms }}, attributes [0] Attributes{{ CRIAttributes }} }

В классе реализованы следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.CERTIFICATIONREQUESTINFO)	Конструктор копированием экземпляра объекта CertificationRequestInfo

СВОЙСТВА

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
SUBJECT	String	Задает значение поля Subject ("/C=US/O=Test/CN=example.com")
PUBKEY	Key	Задает публичный ключ
VERSION	number	Задает номер версии

Пояснения по использованию конструкторов и свойств.

subject(x509name: string)

Метод задает значение поля subject запроса в формате x.509. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
X509NAME	string	Значение в формате x.509

pubkey(pubkey: Key)

Метод задает публичный ключ. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PUBKEY	key	Публичный ключ

version(version: number)

Метод задает публичный ключ. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
VERSION	number	Номер версии

КЛАСС CERTIFICATIONCOLLECTION

Класс реализует представление коллекции «X509» сертификатов. В классе реализованы следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.CERTIFICATECOLLECTION)	Конструктор копированием экземпляра объекта CertificateCollection

СВОЙСТВА

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
LENGTH	number	Возвращает размер коллекции

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
ITEMS	Возвращает элемент из коллекции по заданному индексу
PUSH	Добавляет новый элемент в коллекцию
POP	Удаляет последний элемент их коллекции
REMOVEAT	Удаляет элемент из коллекции по заданному индексу

Пояснения по использованию конструкторов и свойств.

items(index: number): Certificate

Метод возвращает элемент из коллекции по заданному индексу. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require("trusted-crypto");

var certs = new trusted.pki.CertificateCollection();

certs.push(trusted.pki.Certificate.load("test.cert", trusted.DataFormat.PEM));

var cert = certs.items(0); console.log(cert.version); //2
```

push(cert: Certificate): void

Метод добавляет новый элемент в коллекцию. В методе имеется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Элемент для добавления в коллекцию

Например,

```
var trusted = require("trusted-crypto");
var certs = new trusted.pki.CertificateCollection();
certs.push(trusted.pki.Certificate.load("test.crt"));
console.log(certs.length); //1
```

pop(): void

Метод удаляет последний элемент из коллекции. Например,

```
var trusted = require("trusted-crypto");
var certs = new trusted.pki.CertificateCollection();
 certs.push(trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM));
certs.pop();
console.log(certs.length); //0
```

removeAt(index: number): void

Метод удаляет элемент из коллекции по заданному индексу. Метод имеет следующий параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require("trusted-crypto");
var certs = new trusted.pki.CertificateCollection();
 certs.push(trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM));
certs.push(trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM));
certs.removeAt(0);
console.log(certs.length); //1
```

Класс Chain

Класс предназначен для построения цепочки сертификатов (доверия). В классе реализованы следующие конструкторы и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

Методы

МЕТОД	ОПИСАНИЕ
BUILDCHAIN	Выполняет построение цепочки сертификатов
VERIFYCHAIN	Возвращает статус проверки сертификата относительно цепочки

Пояснения по использованию конструкторов и свойств.

buildChain(cert: Certificate, certs: CertificateCollection): CertificateCollection

Метод возвращает коллекцию сертификатов CertificateCollection, участвующих при построении цепочки. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат, для которого производится построение цепочки
CERTS	CertificateCollection	Коллекция сертификатов цепочки

В качестве параметра функции указывается сертификат cert, для которого производится построение цепочки, коллекция сертификатов certs относительно которой происходит построение цепочки.

Например,

```
var trusted = require("trusted-crypto"); var chain = new trusted.pki.Chain();

var certs = new trusted.pki.CertificateCollection();

var cert1 = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.DER); certs.push(cert1);

var cert2 = trusted.pki.Certificate.load("cert.crt", trusted.DataFormat.PEM); certs.push(cert2);

cert3 = trusted.pki.Certificate.load("cert1.crt", trusted.DataFormat.PEM); certs.push(cert3);

var outchain = chain.buildChain(cert2, certs);
```

verifyChain(chain: CertificateCollection, crls: CrlCollection): boolean

Метод выполняет проверку относительно цепочки сертификатов. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CHAIN	CertificateCollection	Коллекция сертификатов цепочки

CRLS	CrlCollection	Коллекция списков отзыва сертификатов

Например,

```
var trusted = require("trusted-crypto"); var chain = new trusted.pki.Chain();
var certs = new trusted.pki.CertificateCollection();
var cert1 = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.DER);
certs.push(cert1);
var cert2 = trusted.pki.Certificate.load("cert.crt", trusted.DataFormat.PEM);
certs.push(cert2);
cert3 = trusted.pki.Certificate.load("cert1.crt", trusted.DataFormat.PEM);
certs.push(cert3);
var outchain = chain.buildChain(cert2, certs);
var crls = new trusted.pki.CrlCollection();
crls.push(trusted.pki.Crl.load("test.crl"));
chain.verifyChain(outChain, crls);
```

Класс Cipher

Класс предназначен для шифрования/расшифрования данных. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR(CIPHERNAME: STRING)	Конструктор параметров в виде наименования алгоритма

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
CRYPTOMETHOD	CryptoMethod	Задает криптографический метод
RECIPIENTSCERTS	CertificateCollection	Задает коллекцию сертификатов получателей
PRIVKEY	Key	Задает закрытый ключ
RECIPIENTCERT	Certificate	Задает сертификат получателя
PASSWORD	string	Задает пароль для шифрования
DIGEST	string	Задает хеш алгоритм
RIV	Buffer	Задает вектор инициализации
IV	string	Возвращает вектор инициализации
RKEY	Buffer	Задает ключ, использующийся для шифрования или расшифрования
KEY	string	Возвращает ключ, использующийся для шифрования или расшифрования
RSALT	Buffer	Задает псевдослучайную вставку
SALT	string	Возвращает псевдослучайную вставку
ALGORITHM	String	Возвращает алгоритм
MODE	String	Возвращает режим шифрования
DGST	String	Возвращает хеш алгоритм

Методы

МЕТОД	ОПИСАНИЕ
ENCRYPT	Реализация шифрования данных
DECRYPT	Реализация расшифрования данных
GETRRECIPIENTINFOS	Возвращает информацию о получателях

Пояснения по использованию конструкторов, свойств и методов.

encrypt(filenameSource: string, filenameEnc: string, format?: DataFormat): void

Метод выполняет шифрование данных. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAMESOURCE	string	Исходное сообщение
FILENAMEENC	String	Зашифрованное сообщение
FORMAT	DataFormat	Формат представления.

Например,

```
var trusted = require("trusted-crypto");
cipher = new trusted.pki.Cipher("aes256");
cipher.cryptoMethod = trusted.CryptoMethod.SYMMETRIC;
cipher.digest = "MD5";
cipher.password = "4321";
 cipher.encrypt("test.txt", "encSym.enc");
```

Следующий пример,

```
var trusted = require("trusted-crypto");
var cipher = new trusted.pki.Cipher("aes256");
var cert = new trusted.pki.CertificateCollection();
cert.push(trusted.pki.Certificate.load("cert.crt", trusted.DataFormat.PEM));
cipher.recipientsCerts = cert;
cipher.encrypt("test.txt", "encAssym.enc", trusted.DataFormat.PEM);
```

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

Метод выполняет расшифрование данных. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAMEENC	string	Шифрованное сообщение
FILENAMEDEC	string	Расшифрованное сообщение
FORMAT	DataFormat	Формат представления. Опционно. По умолчанию - DER

Например,

```
var trusted = require('trusted-crypto');
var cipher = new trusted.pki.Cipher('aes256');
cipher.cryptoMethod = trusted.CryptoMethod.SYMMETRIC;
cipher.digest = "MD5";
cipher.password = "4321";
cipher.decrypt("encSym.enc", "decSym.txt", trusted.DataFormat.PEM);
```

Класс Crl

Класс предназначен для работы со списком отзыва сертификатов в представлении «X509_CRL». В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.CRL)	Конструктор копированием экземпляра объекта CRL

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
ENCODED	Buffer	Возвращает ASN.1 объект CRL в DER кодировке
SIGNATURE	Buffer	Возвращает значение подписи
VERSION	number	Возвращает версию
ISSUERNAME	string	Возвращает имя издателя
ISSUERFRIENDLYNAME	string	Возвращает пользовательское имя издателя сертификата
LASTUPDATE	Date	Возвращает дату последнего обновления
NEXTUPDATE	Date	Возвращает дату следующего обновления
THUMBPRINT	string	Возвращает отпечаток (SHA1)
SIGALGNAME	string	Возвращает имя алгоритма подписи CRL
SIGALGSHORTNAME	string	Возвращает короткое имя алгоритма подписи CRL
SIGALGOID	string	Возвращает OID алгоритма подписи CRL

Методы

МЕТОД	ОПИСАНИЕ
LOAD	чтение структуры из файла
IMPORT	чтение структуры из памяти
EXPORT	сохранение структуры в память
SAVE	сохранение структуры в файл
COMPARE	равнение crl
EQUALS	сравнение
HASH	возвращает хэш структуры по заданному алгоритму
DUPLICATE	создает копию элемента
REVOKED	Возвращает коллекцию списков отзыва

Пояснения по использованию конструкторов, свойств и методов.

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

Метод выполняет чтение структуры из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных.

Например,

```
var trusted = require('trusted-crypto');

crl = new trusted.pki.Crl();

crl.load("certcrl.crl", trusted.DataFormat.DER);

console.log(crl.sigAlgName);
```

static load(filename: string, format: DataFormat): Crl

Методе выполняет чтение структуры из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу
FORMAT	DataFormat	Формат данных.

import(buffer: Buffer, format: DataFormat): void

Метод выполняет чтение структуры из памяти. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
BUFFER	Buffer	Буфер с данными в памяти
FORMAT	DataFormat	Формат данных.

Например,

```
var trusted = require('trusted-crypto');

crl = new trusted.pki.Crl();

crl.load("certcrl.crl", trusted.DataFormat.DER);

var buf = crl.export();

crl.import(buf, trusted.DataFormat.DER);

console.log(crl.version);
```

static import(buffer: Buffer, format: DataFormat): Crl

Метод выполняет чтение структуры из памяти. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
BUFFER	Buffer	Буфер с данными в памяти
FORMAT	DataFormat	Формат данных.

export(format?: DataFormat): Buffer

Метод выполняет сохранение структуры в память. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FORMAT	DataFormat	Формат данных.

Например,

```
var trusted = require('trusted-crypto');

crl = new trusted.pki.Crl();

crl.load("certcrl.crl", trusted.DataFormat.DER);

var buf = crl.export();
```

save(filename: string, dataFormat: DataFormat): void

Метод выполняет сохранение структуры в файл. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	String	Путь к файлу
DATAFORMAT	DataFormat	Формат данных.

Например,

```
var trusted = require('trusted-crypto');

var crl1 = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

 crl1.save ('certcrl2.crl');
```

compare(crl: Crl): number

Метод выполняет сравнение crl. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CRL	Crl	Экземпляр объекта Crl

Например,

```
var trusted = require('trusted-
crypto');

var crl1 = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

 var crl2 = trusted.pki.Crl.load("certcrl1.crl", trusted.DataFormat.DER);

console.log(crl1.compare(crl2)); // 0
```

equals(crl: Crl): boolean

Метод выполняет сравнение crl. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CRL	Crl	Экземпляр объекта Crl

Например,

```
var trusted = require('trusted-
crypto');

var crl1 = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);
```

```
var crl2 = trusted.pki.Crl.load("certcrl1.crl", trusted.DataFormat.DER);
console.log(crl1.equals(crl2)); // 0
```

hash(algorithm?: string): String

Метод возвращает хэш структуры по заданному алгоритму. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
ALGORITHM	string	Наименование алгоритма

Например,

```
var trusted = require('trusted-crypto');
var crl1 = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);
var hash = crl1.hash("sha1");
console.log(hash);
```

duplicate(): Crl

Метод создает копию элемента.

Например,

```
var trusted = require('trusted-crypto');
var crl1 = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);
var crl2 = crl1.duplicate();
console.log(crl1.equals(crl2));
```

revoked(): RevokedCollection

Метод возвращает коллекцию списков отзыва.

Например,

```
var trusted = require('trusted-crypto');
var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);
var rvst = crl.revoked;
consule.log(rvst.length);
```

Класс CrlCollection

Класс предназначен для управления коллекциями списков отзыва сертификатов. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.CRLCOLLECTION)	Конструктор копированием экземпляра объекта CrlCollection

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
LENGTH	number	Размер коллекции (число сертификатов)

Методы

МЕТОД	ОПИСАНИЕ
ITEMS	Возвращает элемент коллекции списков отзыва
PUSH	Добавляет элементы в коллекцию
POP	Удаляет последний элемент из коллекции
REMOVEAT	Удаляет элемент из коллекции

Пояснения по использованию конструкторов, свойств и методов.

items(index: number): Crl

Метод возвращает элемент коллекции списков отзыва. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require('trusted-crypto');

 var crls = new trusted.pki.CrlCollection();

crls.push(trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER));

var crl = crls.items(0); console.log(crl.sigAlgName);
```

push(crl: Crl): void

Метод добавляет элементы в коллекцию. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CRL	Crl	Экземпляр объекта списка отзыва

Например,

```
var trusted = require('trusted-crypto');
var crls = new trusted.pki.CrlCollection();
crls.push(trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER));
console.log(crls.length); //1
```

pop(): void

Метод удаляет последний элемент из коллекции.

removeAt(index: number): void

Метод удаляет элемент из коллекции. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require('trusted-crypto');
var crls = new trusted.pki.CrlCollection();
crls.push(trusted.pki.Crl.load("certcrl.crl" trusted.DataFormat.DER));
crls.removeAt(0);
console.log(crls.length); //0
```

Класс Csr

Класс предназначен для создания запроса на сертификат. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR(NAME: STRING, KEY: KEY, DIGEST: STRING)	

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
ENCODED	Buffer	Получение сформированного запроса на сертификат в Hex представлении

Методы

МЕТОД	ОПИСАНИЕ
SAVE	Сохранение запроса на сертификат

Пояснения по использованию конструкторов, свойств и методов.

save(filename: string, dataFormat: DataFormat): void

Метод выполняет сохранение запроса на сертификат. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Полный путь к файлу
DATAFORMAT	DataFormat	Формат данных

Например,

```
var trusted = require("trusted-crypto");
var key = trusted.pki.Key.readPrivateKey("cert.key", trusted.DataFormat.PEM, "");
var csr = trusted.pki.CSR("/C=US/O=Test/CN=example.com", key, "SHA1");
csr.save("test.csr", trusted.DataFormat.DER);
```

Класс Key

Класс предназначен для генерации ключевой пары. Ключевая пара создается в виде структуры:

KEYPAIR{KeyAlgorithm algorithm; EVP_PKEY pkey;} enum KeySize{1024, 2048, 4096}

В классе реализованы следующие конструкторы и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.KEY)	Конструктор копированием экземпляра объекта Key

Методы

МЕТОД	ОПИСАНИЕ
GENERATE	Генерация ключевой пары
READPRIVATEKEY	Чтение приватного ключа из файла
WRITEPRIVATEKEY	Запись приватного ключа в файл
READPUBLICKEY	Чтение открытого ключа из файла
WRITEPUBLICKEY	Запись открытого ключа в файл
COMPARE	Сравнение ключей

Пояснения по использованию конструкторов, свойств и методов.

generate(format: DataFormat, pubExp: PublicExponent, keySize: number, password: string): Key

Метод выполняет генерацию ключевой пары. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER
PUBEXP	PublicExponent	Значениеоткрытой экспоненты
KEYSIZE	number	Размер ключа
PASSWORD	string	Пароль для доступа к шифрованному контенту

Например,

```
var trusted = require('trusted-crypto');
var key = new trusted.pki.Key();
var keyPair = key.generate(trusted.DataFormat.PEM, trusted.PublicExponent.RSA_F4, 1024);
```

readPrivateKey(filename: string, format: DataFormat, password: string): Key

Метод выполняет чтение приватного ключа из файла. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER
PASSWORD	string	Пароль для доступа к шифрованному контенту

Например,

var trusted = require('trusted-crypto'); var key = new trusted.pki.Key();

var privateKey = key.readPrivateKey("privkey_s.key", trusted.DataFormat.PEM, "1234");

writePrivateKey(filename: string, format: DataFormat, password: string): any

Метод выполняет запись приватного ключа в файл. Метод использует параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER
PASSWORD	string	Пароль для доступа к шифрованному контенту

Например,

var trusted = require('trusted-crypto'); var key = new trusted.pki.Key();

var keyPair = key.generate(trusted.DataFormat.PEM, trusted.PublicExponent.RSA_F4, 1024);
keyPair.writePrivateKey("privkey_s.key", trusted.DataFormat.PEM, "1234");

readPublicKey(filename: string, format: DataFormat): Key

Метод выполняет чтение открытого ключа из файла. Метод использует параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	путь к файлу
FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER

Например,

var trusted = require('trusted-crypto');

var key = new trusted.pki.Key();

publickey = key.readPublicKey("pubkey_s.key", trusted.DataFormat.PEM);

writePublicKey(filename: string, format: DataFormat): any

Метод выполняет запись открытого ключа в файл. Метод использует параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	путь к файлу

FORMAT	DataFormat	Формат данных. Опционально. По умолчанию DER

Например,

var trusted = require('trusted-crypto'); var key = new trusted.pki.Key();

var keyPair = key.generate(trusted.DataFormat.PEM, trusted.PublicExponent.RSA_F4, 1024); keyPair.writePublicKey("pubkey_s.key", trusted.DataFormat.PEM);

compare(key: Key): number

Метод выполняет сравнение ключей. Метод использует параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
KEY	Key	Экземпляр объекта Key

Например,

var trusted = require('trusted-crypto'); var key = new trusted.pki.Key();

var privateKey = key.readPrivateKey("privkey_s.key", trusted.DataFormat.PEM, "1234");

var privateKey1 = key.readPrivateKey("privkey_s.key", trusted.DataFormat.PEM, "1234");

privateKey.compare(privateKey1);

Класс Oid

Класс предназначен для преобразования объектного идентификатора (OID'a). В классе реализованы следующие конструкторы и свойства:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR(OID: STRING)	Конструктор по умолчанию
CONSTRUCTOR(HANDLE: NATIVE.PKI.OID)	Конструктор копированием экземпляра объекта Key

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
VALUE	string	Возвращает текстовое представление значения OID
LONGNAME	string	Возвращает длинное имя OID
SHORTNAME	string	Возвращает короткое имя OID

Например,

```
var oid = new trusted.Pki.Oid("2.5.4.3");
console.log(oid.longName);      // commonName
console.log(oid.shortName);     // CN
console.log(oid.value);         // 2.5.4.3
```

Класс Pkcs12

Класс предназначен для работы с форматом PKCS#12 для извлечения закрытого ключа и сертификата. В классе реализованы следующие конструкторы и свойства:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(param?: NATIVE.PKI.PKCS12)	Конструктор копированием экземпляра объекта Pkcs12

Методы

МЕТОД	ОПИСАНИЕ
CERTIFICATE	Возвращает сертификат
KEY	Возвращает приватный ключ
CA	Возвращает цепочку сертификатов
LOAD	*Чтение pkcs12 из файла*
SAVE	*Сохранение pkcs12 в файл*
CREATE	*Создание структуры PKCS12*

Пояснения по использованию конструкторов и методов.

certificate(password: string): Certificate

Метод возвращает сертификат. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PASSWORD	string	Пароль доступа к шифрованному контенту

Например,

```
var trusted = require('trusted-crypto'); var p12 = new trusted.pki.Pkcs12();
p12.load("test.pfx");
var cert = p12.certificate("password");
```

key(password: string): Key

Метод возвращает приватный ключ. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PASSWORD	string	Пароль доступа к шифрованному контенту

Например,

```
var trusted = require('trusted-crypto'); var p12 = new trusted.pki.Pkcs12(); p12.load("test.pfx");
var key = p12.key("password");
```

ca(password: string): CertificateCollection

Метод возвращает цепочку сертификатов. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PASSWORD	string	Пароль доступа к шифрованному контенту

Например,

```
var trusted = require('trusted-crypto');
var p12 = new trusted.pki.Pkcs12();
p12.load("test.pfx");
var ca = p12.ca("password");
console.log (ca.length);
```

load(filename: string): void

Метод выполняет чтение pkcs12 из файла. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу

Например,

```
var trusted = require('trusted-crypto');
var p12 = new trusted.pki.Pkcs12();
p12.load("test.pfx");
```

static load(filename: string): Pkcs12

Метод выполняет чтение сертификата из файла. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу

save(filename: string): void

Сохранение pkcs12 в файл. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	string	Путь к файлу

Например,

```
var trusted = require('trusted-crypto');
```

```
var p12 = new trusted.pki.Pkcs12();
var cert = trusted.pki.Certificate.load("./test/cert.crt", trusted.DataFormat.PEM);
var key = trusted.pki.Key.readPrivateKey("./test/cert.key", trusted.DataFormat.PEM, "");
var p12Res = p12.create(cert, key, null, "1", "test_name");
p12Res.save('test_pkcs12.pfx');
```

create(cert: Certificate, key: Key, ca: CertificateCollection, password: string, name: string): Pkcs12

Метод выполняет создание структуры PKCS12. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат
KEY	Key	Приватный ключ
CA	CertificateCollection	Цепочка сертификатов
PASSWORD	String	Пароль доступа к шифрованному контенту
NAME	string	Пользовательское имя

Например,

```
var trusted = require('trusted-crypto');
var p12 = new trusted.pki.Pkcs12();
var cert = trusted.pki.Certificate.load("./test/cert.crt", trusted.DataFormat.PEM);
var key = trusted.pki.Key.readPrivateKey("./test/cert.key", trusted.DataFormat.PEM, "");
var p12Res = p12.create(cert, key, null, "1", "test_name");
```

Класс Revocation

Класс предназначен для проверки наличия и получения (загрузки) CRL (аналог Revocation- провайдера). В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

Методы

МЕТОД	ОПИСАНИЕ
GETCRLLOCAL	Ищет crl для сертификата в локальном хранилище
GETCRLDISTPOINTS	Возвращает массив из точек распространения crl
CHECKCRLTIME	Проверяет действительность времени crl
DOWNLOADCRL	Загружает crl

Пояснения по использованию конструкторов, свойств и методов.

getCrlLocal(cert: Certificate, store: PkiStore): any

Метод ищет crl для сертификата в локальном хранилище. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат
STORE	PkiStore	Хранилище

Например,

```
var trusted = require('trusted-crypto');

var revocation = new trusted.pki.Revocation();

 var cert = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM);

var providerSystem = new trusted.pkistore.Provider_System(“/test/CertStore”);

var store = new trusted.pkistore.PkiStore("/test/CertStore/cash.json");

var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

store.addCrl(providerSystem.handle, "CRL", crl, 0);

store.addProvider(providerSystem.handle);

var tmpCrl = revocation.getCrlLocal(cert, store);
```

getCrlDistPoints(cert: Certificate): Array<string>

Метод возвращает массив из точек распространения crl. В методе используются

параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат

Например,

```
var trusted = require('trusted-crypto');

var revocation = new trusted.pki.Revocation();

var cert = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM);

var array = revocation.getCrlDistPoints(cert);
```

checkCrlTime(crl: Crl): boolean

Метод проверяет действительность времени crl. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CRL	Crl	Список отозванных сертификатов

Например,

```
var trusted = require('trusted-crypto');

var revocation = new trusted.pki.Revocation();

var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

revocation.checkCrlTime(crl);
```

downloadCRL(distPoints: Array<string>, pathForSave: string, done: Function): void

Метод загружает crl. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
DISTPOINTS	Array<string>	Точки распространения crl
PATHFORSAVE	String	Путь к файлу
DONE	Function	Функция обратного вызова

Например,

```
var trusted = require('trusted-crypto');

var revocation = new trusted.pki.Revocation();

var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

var cert = trusted.pki.Certificate.load("test.crt", trusted.DataFormat.PEM);

var distPoints = revocation.getCrlDistPoints(cert);

revocation.downloadCRL(distPoints, "/test/temp.crl", function(err, res) { });
```

Класс Revoked

Класс предназначен для получения расширений списка отозванных сертификатов. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(param?: native.PKI.Revoked)	Конструктор экземпляра объекта Revoked

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
SERIALNUMBER	String	Возвращает серийный номер отозванного сертификата
REVOCATIONDATE	String	Возвращает дату отзыва
REASON	String	Возвращает причину отзыва

методы

МЕТОД	ОПИСАНИЕ
DUPLICATE	Создает копию расширений

Пояснения по использованию конструкторов, свойств и методов.

duplicate(): Revoked

Метод создает копию расширений.

Например,

```
var trusted = require('trusted-crypto');

var assert = require(‘assert’);

var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

var rvst = crl.revoked;

var rv = rvst.items(0);

var rv1 = rv.duplicate();

assert.equal(rv.serialNumber == rv1.serialNumber, true);

assert.equal(rv.revocationDate == rv1.revocationDate, true);

assert.equal(rv.reason == rv1.reason, true);
```

Класс Revokeds

Класс предназначен для управления коллекциями расширений отозванных сертификатов. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию
CONSTRUCTOR(param?: native.PKI.RevokedCollection)	Конструктор экземпляра объекта Revokeds

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
LENGTH	Number	Возвращает размер коллекции

Методы

МЕТОД	ОПИСАНИЕ
ITEMS	Возвращает элемент коллекции
PUSH	Добавляет элементы в коллекцию
POP	Удаляет последний элемент из коллекции
REMOVEAT	Удаляет элемент из коллекции

Пояснения по использованию конструкторов, свойств и методов.

items(index: number): Revoked

Метод возвращает элемент коллекции расширений списков отзыва. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require('trusted-crypto');

var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

var rvst = crl.revoked;

var rv = rvst.items(0);
```

push(rv: Revoked): void

Метод добавляет элементы в коллекцию. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
rv	Revoked	Экземпляр объекта расширений списка отзыва

Например,

```
var trusted = require('trusted-crypto');

var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

var rvst = crl.revoked;

var rv = rvst.items(0);

rvst.push(rv);
```

pop(): void

Метод удаляет последний элемент из коллекции.

Например,

```
var trusted = require('trusted-crypto');

var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

var rvst = crl.revoked;

var rv = rvst.items(0);

rvst.pop();
```

removeAt(index: number): void

Метод удаляет элемент из коллекции. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
INDEX	number	Индекс элемента в коллекции

Например,

```
var trusted = require('trusted-crypto');

var crl = trusted.pki.Crl.load("certcrl.crl", trusted.DataFormat.DER);

var rvst = crl.revoked;

var rv = rvst.items(0);

rvst.removeAt(0);
```

3.3. ПАКЕТ PKISTORE

КЛАСС CASHJSON

Класс предназначен для создания, загрузки и сохранения кэша хранилищ объектов PKI в виде JSON. В классе реализованы следующие конструкторы и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR(FILENAME: STRING)	Конструктор с параметром пути к файлу JSON

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
EXPORT	Выполняет экспорт из файла кэша
IMPORT	Выполняет импорт в файл кэша

Пояснения по использованию конструкторов, свойств и методов.

export(): native.PKISTORE.IPkiItem[]

Метод выполняет экспорт из файла кэша. Например,

```
var trusted = require('trusted-crypto');
cashjson = new trusted.pkistore.CashJson('CertStore/cash.json');
var items = cashjson.export();
console.log(items.length);
```

import(items: native.PKISTORE.IPkiItem[]): void

Метод выполняет импорт в файл кэша. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
ITEMS	native.PKISTORE.IPkiItem[]	Массив описателей объектов PKI

Например,

```
var trusted = require('trusted-crypto');
cashjson = new trusted.pkistore.CashJson('CertStore/cash.json');
var items = cashjson.export();
cashjson.import(items);
```

Класс ProviderCryptopro

Класс предназначен для создания провайдера КриптоПро CSP. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

Методы

МЕТОД	ОПИСАНИЕ
GETKEY	Возвращает закрытый ключ сертификата из хранилища КриптоПро
HASPRIVATEKEY	Возвращает true, если у сертификата доступен закрытый ключ

getKey(cert: pki.Certificate)

Метод возвращает закрытый ключ сертификата из хранилища КриптоПРО. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат, к которому привязывается закрытый ключ

Например,

```
var trusted = require('trusted-crypto');

var cert = trusted.pki.Certificate.load(‘cert.crt’, trusted.DataFormat.PEM);

var providerCryptopro = new trusted.pkistore.ProviderCryptopro();

var key = providerCryptopro.getKey(cert);
```

hasPrivateKey(cert: pki.Certificate)

Метод проверяет, доступен ли закрытый ключ сертификата. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат, для которого проверяется доступность закрытого ключа

Например,

```
var trusted = require('trusted-crypto');

var cert = trusted.pki.Certificate.load(‘cert.crt’, trusted.DataFormat.PEM);

var providerCryptopro = new trusted.pkistore.ProviderCryptopro();

providerCryptopro.hasPrivateKey(cert);
```

Класс Filter

Класс предназначен для поиска и предоставления основных атрибутов объектов PKI. В классе реализованы следующие конструкторы и свойства:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
TYPES	string	Возвращает тип объекта
PROVIDERS	string	Возвращает провайдер хранилища в котором размещен объект
CATEGORYS	string	Возвращает категорию хранилища (MY, OTHER, TRUST)
HASH	String	Возвращает хэш объекта
SUBJECTNAME	string	Возвращает полное имя владельца сертификата
SUBJECTFRIENDLYNAME	string	Возвращает пользовательское имя владельца сертификата
ISSUERNAME	string	Возвращает полное имя издателя сертификата
ISSUERFRIENDLYNAME	String	Возвращает пользовательское имя издателя сертификата
SERIAL	string	Возвращает серийный номер сертификата

Класс PKIItem

Класс предназначен для задания атрибутов-описателей PKI объектов. В классе реализованы следующие конструкторы и свойства:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
FORMAT	string	Задает формат представления объекта
TYPE	string	Задает тип объекта
PROVIDER	string	Задает провайдер хранилища в котором размещен объект
CATEGORY	string	Задает категорию хранилища (MY, OTHER, TRUST)
URI	string	Задает URI к физическому месторасположению объекта
HASH	string	Задает хэш объекта
SUBJECTNAME	string	Задает полное имя владельца сертификата
SUBJECTFRIENDLYNAME	string	Задает пользовательское имя владельца сертификата
ISSUERNAME	string	Задает полное имя издателя сертификата
ISSUERFRIENDLYNAME	string	Задает пользовательское имя издателя сертификата
SERIAL	string	Задает серийный номер сертификата
NOTBEFORE	String	Задает дату начала действия сертификата
NOTAFTER	string	Задает дату истечения срока действия сертификата
LASTUPDATE	string	Задает дату последнего обновления списка отзыва
NEXTUPDATE	string	Задает дату следующего выпуска списка отзыва
KEY	string	Задает хеш публичного ключа
KEYENC	boolean	Задает зашифрован ли закрытый ключ
ORGANIZATIONNAME	string	Задает название организации
SIGNATUREALGORITHM	string	Задает алгоритм подписи

Класс PkiStore

Класс предназначен для организации стека хранилищ сертификатов и ключей. Создается кэш (JSON) для формирования описателей объектов хранения. В классе реализованы следующие конструкторы, свойства и методы:

Конструкторы

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR(FOLDER: STRING)	Конструктор по умолчанию
CONSTRUCTOR(PARAM: NATIVE.PKISTORE.PKISTORE)	Конструктор копированием экземпляра объекта PkiStore

Свойства

НАЗВАНИЕ	ТИП	ОПИСАНИЕ
CASH	CashJson	Возвращает структуру кэша (JSON)

Методы

МЕТОД	ОПИСАНИЕ
ADDPROVIDER	Добавляет новый провайдер хранилища
ADDCERT	Добавляет описатель сертификата
ADDCRL	Добавляет описатель списка отзыва
ADDKEY	Добавляет описатель ключа
ADDCSR	Добавляет описатель запроса на сертификат
FIND	Поиск объекта в кэше
FINDKEY	Поиск ключа в кэше
GETITEM	Извлечение объекта по его описателю
CERTS	Возвращает коллекцию сертификатов в хранилищах

Пояснения по использованию конструкторов, свойств и методов.

addProvider(provider: native.PKISTORE.Provider): void

Метод добавляет новый провайдер хранилища. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PROVIDER	native.PKISTORE.Provider	Провайдер хранилища

Например,

```
var providerSystem = new trusted.pkistore.Provider_System('/CertStore');
store.addProvider(providerSystem.handle);
```

addCert(provider: native.PKISTORE.Provider, category: string, cert: Certificate): string

Метод добавляет описатель сертификата. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PROVIDER	native.PKISTORE.Provider	Провайдер хранилища
CATEGORY	string	Категория хранилища (MY, OTHER, TRUST)
CERT	Certificate	Сертификат для которого создается описатель

Например,

```
var trusted = require('trusted-crypto');

var providerSystem = new trusted.pkistore.Provider_System('/CertStore');

var store = new trusted.pkistore.PkiStore("/CertStore/cash.json");

var cert = trusted.pki.Certificate.load("cert1.crt", trusted.DataFormat.PEM);
store.addCert(providerSystem.handle, "MY", cert);
```

addCrl(provider: native.PKISTORE.Provider, category: string, crl: Crl): string

Метод добавляет описатель списка отзыва. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PROVIDER	native.PKISTORE.Provider	Провайдер хранилища
CATEGORY	string	Категория хранилища (MY, OTHER, TRUST)
CRL	Crl	Список отзыва для которого создается описатель

Например,

```
var trusted = require('trusted-crypto');

var providerSystem = new trusted.pkistore.Provider_System('/CertStore');

var store = new trusted.pkistore.PkiStore("CertStore/cash.json");

crl = trusted.pki.Crl.load("certcrl.crl"); store.addCrl(providerSystem.handle, "CRL", crl);
```

addKey(provider: native.PKISTORE.Provider, key: Key, password: string): string

Метод добавляет описатель ключа. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PROVIDER	native.PKISTORE.Provider	Провайдер хранилища
KEY	Key	Ключ для которого создается описатель
PASSWORD	string	Пароль для доступа к шифрованному контенту

Например,

```
var trusted = require('trusted-crypto');

var providerSystem = new trusted.pkistore.Provider_System('CertStore');

var store = new trusted.pkistore.PkiStore("CertStore/cash.json");

var key = trusted.pki.Key.readPrivateKey("cert.key", trusted.DataFormat.PEM, "");

store.addKey(providerSystem.handle, key, "password");
```

addCsr(provider: native.PKISTORE.Provider, category: string, csr: CertificationRequest): string

Метод добавляет описатель запроса на сертификат. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ

PROVIDER	native.PKISTORE.Provider	Провайдер хранилища
CATEGORY	string	Категория хранилища (MY, OTHER, TRUST)
CSR	CertificationRequest	Запрос на сертификат для которого создается описатель

Например,

```
var trusted = require('trusted-crypto');
var providerSystem = new trusted.pkistore.Provider_System('/CertStore');
var store = new trusted.pkistore.PkiStore("/CertStore/cash.json");
var csr = trusted.pki.CertificationRequest.load("test.csr", trusted.DataFormat.PEM, "");
store.addCsr(providerSystem.handle, "MY", csr);
```

find(ifilter?: native.PKISTORE.IFilter): native.PKISTORE.IPkiItem[]

Метод выполняет поиск объекта в кэше. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
IFILTER	native.PKISTORE.IFilter	Фильтра для выборки объектов PKI

Например,

```
var trusted = require('trusted-crypto');
var providerSystem = new trusted.pkistore.Provider_System('CertStore');
var store = new trusted.pkistore.PkiStore("CertStore/cash.json");
store.addProvider(providerSystem.handle);
var items = store.find({type: ["CERTIFICATE"], category: ["MY"]});
console.log(items.length);
```

findKey(ifilter: native.PKISTORE.IFilter): native.PKISTORE.IPkiItem

Метод выполняет поиск ключа в кэше. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
IFILTER	native.PKISTORE.IFilter	Фильтра для выборки объектов PKI

Например,

```
var trusted = require('trusted-crypto');
var providerSystem = new trusted.pkistore.Provider_System('CertStore');
var store = new trusted.pkistore.PkiStore("CertStore/cash.json");
store.addProvider(providerSystem.handle);
var key = store.findKey({ type: ["CERTIFICATE"],
```

provider: ["SYSTEM"], category: ["MY"], hash: ['67cd1d796cfb42d00166737c6e16d596cf83695e']});

console.log(key.uri);

getItem(item: native.PKISTORE.IPkiItem): any

Метод выполняет извлечение объекта по его описателю. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
ITEM	native.PKISTORE.IPkiItem	Описатель объекта

Например,

var trusted = require('trusted-crypto');

var providerSystem = new trusted.pkistore.Provider_System('CertStore');

var store = new trusted.pkistore.PkiStore("CertStore/cash.json");

store.addProvider(providerSystem.handle);

var certs = store.find({ type: ["CERTIFICATE"], category: ["MY"]});

var item = certs[0];

var object = store.getItem(item); console.log(object.type);

certs():pki.CertificateCollection

Метод выполняет поиск ключа в кэше. В методе используется параметр

Например,

var trusted = require('trusted-crypto');

var providerSystem = new trusted.pkistore.Provider_System('CertStore');

var store = new trusted.pkistore.PkiStore("CertStore/cash.json");

store.addProvider(providerSystem.handle);

var cert = trusted.pki.Certificate.load("cert1.crt", trusted.DataFormat.PEM);

store.addCert(providerSystem.handle, "MY", cert);

var certs = trusted.pkistore.certs();

КЛАСС PROVIDERMICROSOFT

Класс предназначен для создания провайдера Microsoft (только для Windows). В классе реализованы следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
GETKEY	Возвращает закрытый ключ сертификата
HASPRIVATEKEY	Возвращает true, если у сертификата доступен закрытый ключ

Пояснения по использованию конструкторов, свойств и методов.

getKey(cert: pki.Certificate)

Метод возвращает закрытый ключ сертификата. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат, к которому привязывается закрытый ключ

Например,

```
var cert = trusted.pki.Certificate.load('cert.cer', trusted.DataFormat.PEM);

var providerMicrosoft = new trusted.pkistore.ProviderMicrosoft();

var key = providerMicrosoft.getKey(cert);
```

hasPrivateKey(cert: pki.Certificate): boolean

Метод проверяет, доступен ли закрытый ключ для сертификата. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
CERT	Certificate	Сертификат, для которого проверяется доступность закрытого ключа

Например,

```
var cert = trusted.pki.Certificate.load('cert.cer', trusted.DataFormat.PEM);

var providerMicrosoft = new trusted.pkistore.ProviderMicrosoft();

providerMicrosoft.hasPrivateKey(cert);
```

КЛАСС PROVIDER_SYSTEM

Класс предназначен для создания системного провайдера (провайдера OpenSSL). В классе реализованы следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR(FOLDER: STRING)	Конструктор по умолчанию

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
OBJECTTOPKIITEM	Возвращает PKI объект

Пояснения по использованию конструкторов, свойств и методов.

objectToPkiItem(path: string): native.PKISTORE.IPkiItem

Метод возвращает объект PkiItem. В методе используется параметр

ПАРАМЕТР	ТИП	ОПИСАНИЕ
PATH	string	Задает URI к физическому месторасположению объекта

Например,

```
var trusted = require('trusted-crypto');
var providerSystem = new trusted.pkistore.Provider_System(‘/CertStore’);
var store = new trusted.pkistore.PkiStore( "/CertStore/cash.json");
var uri = store.addCert(providerSystem.handle, "MY", cert);
var item = providerSystem.objectToPkiItem(uri);
```

3.4. ПАКЕТ COMMON

КЛАСС OPENSSL

Класс предназначен как вспомогательный для пакета OpenSSL. В классе реализованы следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
RUN	Загружает engines и добавляет алгоритмы
STOP	Очищает объекты openssl и ошибки
PRINTERRORS	Печатает стек ошибок openssl

Пояснения по использованию конструкторов, свойств и методов.

run(): void

Метод загружает engines и добавляет алгоритмы.

Например,

```
var trusted = require('trusted-crypto');

trusted.common.OpenSSL.run();
```

stop(): void

Метод очищает объекты openssl и ошибки.

Например,

```
var trusted = require('trusted-crypto');

trusted.common.OpenSSL.stop();

trusted.common.OpenSSL.printErrors();
```

stop(): void

Метод печатает стек ошибок openssl.

Например,

```
var trusted = require('trusted-crypto');

trusted.common.OpenSSL.run();

trusted.pki.Certificate.load("undefined");

trusted.common.OpenSSL.printErrors();
```

3.5. ПАКЕТ UTILS

Функция download

Функция предназначена для загрузки файла

ПАРАМЕТРЫ

ПАРАМЕТР	ТИП	ОПИСАНИЕ
URL	string	Задает URL к физическому месторасположению объекта
PATH	string	Путь для сохранения файла
DONE	function	callback

КЛАСС JWT

Класс предназначен для проверки лицензии в формате jwt. В классе реализованы следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
CHECKLICENSE	Проверяет лицензию в формате jwt

Пояснения по использованию конструкторов, свойств и методов.

checkLicense(data?: string): number

Метод возвращает код ошибки при проверке лицензии (возвращает 0, если лицензия корректна).

Например,

```
var trusted = require('trusted-crypto');

trusted.ytils.Jwt.checkLicense();
```

КЛАСС LOGGER

Класс предназначен для управления лог файлом. В классе реализованы следующие конструкторы, свойства и методы:

КОНСТРУКТОРЫ

КОНСТРУКТОР	ОПИСАНИЕ
CONSTRUCTOR()	Конструктор по умолчанию

МЕТОДЫ

МЕТОД	ОПИСАНИЕ
START	Начинает запись логов в файл
STOP	Останавливает запись логов в файл
CLEAR	Очищает существующий лог файл

Пояснения по использованию конструкторов, свойств и методов.

static start(filename: string, level: LoggerLevel = DEFAULT_LOGGER_LEVEL): Logger

Метод начинает запись логов в файл. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	filename	Файл для записи логов
LOGGERLEVEL	level=DEFAULT_LOGGER_LEVEL	Уровень логирования

Например,

```
var trusted = require('trusted-crypto');
logger = trusted.utils.Logger.start("logger.txt", trusted.LoggerLevel.ALL);
```

start(filename: string, level: LoggerLevel = DEFAULT_LOGGER_LEVEL): viod

Метод начинает запись логов в файл. В методе используются параметры

ПАРАМЕТР	ТИП	ОПИСАНИЕ
FILENAME	filename	Файл для записи логов
LOGGERLEVEL	level=DEFAULT_LOGGER_LEVEL	Уровень логирования

Например,

```
var trusted = require('trusted-crypto');
var logger = new trusted.utils.Logger();
logger.start("logger.txt", trusted.LoggerLevel.ALL);
```

stop(): viod

Метод останавливает запись логов в файл.

Например,

```
var trusted = require('trusted-crypto');

logger = trusted.utils.Logger.start("logger.txt", trusted.LoggerLevel.ALL);

logger.stop();
```

clear(): viod

Метод очищает лог файл.

Например,

```
var trusted = require('trusted-crypto');

logger = trusted.utils.Logger.start("logger.txt", trusted.LoggerLevel.ALL);

logger.clean();
```

3.6. Определения

CryptoMethod {

SYMMETRIC = 0,

ASSYMETRIC = 1,

}

DataFormat {

DER = 0,

PEM = 1,

}

PublicExponent {

RSA_3 = 0,

RSA_F4 = 1,

}

LoggerLevel {

NULL = 0,

ERROR = 1,

WARNING = 2,

INFO = 4,

DEBUG = 8,

TRACE = 16,

OPENSSL = 32,

ALL = ERROR | WARNING | INFO | DEBUG | TRACE | OPENSSL

}

Команда разработки и сопровождения продукта

Селедкин Андрей Евгеньевич

Менеджер по маркетингу, andrey.selyodkin@digt.ru

Компетенции в рамках проекта: изучение узкого сегмента рынка программных продуктов, формирование стратегии развития продукта, организация испытаний на совместимость продукта, вывод продукта на рынок, презентация продукта.

Чесноков Сергей Евгеньевич

Инженер-программист, shesnokov@gmail.com

Компетенции в рамках проекта: планирование процесса разработки продукта. разработка графического пользовательского интерфейса продукта, разработка ядра продукта, сборка продукта для различных платформ, создание технической и пользовательской документации, техническая поддержка продукта

Гаврилов Александр Владимирович

Инженер-программист, alg@digt.ru

Компетенции в рамках проекта: разработка графического пользовательского интерфейса, разработка внешних модулей для криптографических преобразований, интеграция с криптопропровайдерами, сопровождение репозиториев OpenSource-частей проекта, техническая поддержка продукта.

Шалагина Наталья Владимировна

Специалист по тестированию, nsh@digt.ru

Компетенции в рамках проекта: разработка методик тестирования продукта под различными платформами, создание технической и пользовательской документации, техническая поддержка продукта.

Контактная информация

Компания «Цифровые технологии» – российский разработчик и поставщик программного обеспечения в области защиты информации, телекоммуникаций и Интернет-сервисов.

Направление исследований и создания программных продуктов:

- разработка кроссплатформенных решений в области защиты данных, как в виде отдельных собственных продуктов, так и технологических стеков.
- встраивание российских сертифицированных криптографических алгоритмов в информационные системы, независимо от их бизнес-задачи.
- создание систем авторизации и аутентификации пользователей.
- консалтинг в области использования средств криптографической защиты информации (СКЗИ) в государственной и коммерческой среде.

Особое внимание разработчики компании уделяют внедрению и поддержки отечественных стандартов защиты информации, в том числе сертифицированных продуктов.

В случае необходимости получения дополнительной информации по продукту КриптоАРМ ГОСТ, можно обратиться непосредственно к разработчикам продукта или в службу технической поддержки компании – support@trusted.ru.

Контактная информация:

info@trusted.ru

8 (8362) 33-70-50, 8 (499) 705-91-10, 8 (800) 555-65-81

424033, РМЭ, г. Йошкар-Ола, ул. Петрова, д.1, а/я 67