127018, Москва, Сущёвский Вал, 18
Телефон: (495) 995 4820
Факс: (495) 995 4820
https://CryptoPro.ru
E-mail: info@CryptoPro.ru

УТВЕРЖДЕНЫ
ЖТЯИ.00091-04 92 01-ЛУ

Средство Криптографической Защиты Информации

КриптоПро JCP

Версия 2.0 R4

Правила пользования

ЖТЯИ.00091-04 92 01
Листов 41

2020 г

Содержание

Аннотация

Данный документ содержит правила пользования средства криптографической защиты информации (СКЗИ) КриптоПро JCP версия 2.0 R4, его состав, назначение, ключевую систему, требования по защите от НСД.

Документ предназначен для администраторов информационной безопасности учреждений, осуществляющих установку, обслуживание контроль за соблюдением требований к эксплуатации средств СКЗИ, а также администраторам серверов, сетевых ресурсов предприятия и других работников службы информационной безопасности, осуществляющим настройку рабочих мест для работы со средствами СКЗИ.

Инструкции администраторам безопасности и пользователям различных автоматизированных систем, использующих СКЗИ КриптоПро JCP версии 2.0 R4, должны разрабатываться с учетом требований настоящих Правил.

1 Назначение СКЗИ. Условия эксплуатации

СКЗИ КриптоПро JCP версии 2.0 R4 предназначено для защиты открытой информации в информационных системах общего пользования (вычисление и проверка ЭП) и защиты конфиденциальной информации, не содержащей сведений, составляющих государственную тайну, в корпоративных информационных системах (шифрование и расшифрование информации, вычисление и проверка имитовставки, вычисление значения хэш-функции, вычисление и проверка ЭП).

СКЗИ КриптоПро JCP версии 2.0 R4 совместимо с СКЗИ КриптоПро CSP по выполняемым криптографическим функциям и ключам.

При эксплуатации СКЗИ КриптоПро JCP версии 2.0 R4 должны выполняться следующие требования:

- Средствами СКЗИ не допускается обрабатывать информацию, содержащую сведения, составляющие государственную тайну.
- Допускается использование СКЗИ для криптографической защиты персональных данных.
- Ключевая информация является конфиденциальной.
- Размещение СКЗИ в помещениях, предназначенных для ведения переговоров, в ходе которых обсуждаются вопросы, содержащие сведения, составляющие государственную тайну, осуществляется установленным порядком.
- ПЭВМ, на которых используется СКЗИ, должны быть допущены для обработки конфиденциальной информации по действующим в Российской Федерации требованиям по защите информации от утечки по техническим каналам, в том числе, по каналу связи (например, СТР-К), с учетом модели угроз в информационной системе заказчика, которым должно противостоять СКЗИ.
- Инсталляция СКЗИ на рабочих местах должна производиться только с дистрибутива, полученного по доверенному каналу.
- Использование СКЗИ для защиты речевой информации без проведения соответствующих дополнительных исследований запрещено. При ведении переговоров, содержащих конфиденциальную информацию, должны отключаться устройства преобразования речи аппаратной компоненты СКЗИ.
- Встраивание СКЗИ в другие средства возможно только с использованием функций, указанных в Приложении 2. Использование при встраивании функций классов, реализованных в сторонних библиотеках или использующих их (например, функций класса XMLSignature), возможно только с использованием библиотек, входящих в дистрибутив СКЗИ КриптоПро JCP версии 2.0 R4. В случае использования прочих вызовов необходимо производить разработку отдельного СКЗИ в соответствии с действующей нормативной базой (в частности, с Постановлением Правительства Российской Федерации от 16 апреля 2012 г. № 313 и Положением о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации (Положение ПКЗ-2005)).

Следующие программные компоненты СКЗИ КриптоПро JCP версии 2.0 R4 можно использовать без проведения дополнительных тематических исследований:

- приложение для выработки CMS сообщений `cmsutil` (см. Приложении 3);
- приложение для создания TLS-туннеля `tls_proxy` (при соблюдении требований Приложения 4).

СКЗИ КриптоПро JCP версии 2.0 R4 можно эксплуатировать без проведения исследований по оценке влияния с веб-сервером в составе ПО Apache Tomcat (Tomcat7.0, Tomcat 8.5, Tomcat 9.0) при условии выполнения требований п. 7.4 ЖТЯИ.00091-04 33 03. Руководство программиста (JTLS).

Использование СКЗИ КриптоПро JCP версии 2.0 R4 с выключенным режимом усиленного контроля использования ключей допускается только в целях тестирования. Включение данного режима описано в разд. 3.6.1.

СКЗИ КриптоПро JCP версии 2.0 R4 при условии выполнения настоящих Правил обеспечивает криптографическую защиту конфиденциальной информации от внешнего нарушителя, самостоятельно осуществляющего создание методов и средств реализации атак, а также самостоятельно реализующего атаки.

2 Порядок распространения СКЗИ

Установочные модули СКЗИ КриптоПро JCP и комплект эксплуатационной документации к нему могут поставляться пользователю Уполномоченной организацией двумя способами:

1) на носителе (CD, DVD-диски);
2) посредством загрузки через Интернет.

Для получения возможности загрузки установочных модулей СКЗИ и комплекта эксплуатационной документации пользователь направляет свои учетные данные Уполномоченной организации. Учетные данные могут быть направлены посредством заполнения специализированной регистрационной формы на сайте Уполномоченной организации.

После получения Уполномоченной организацией учетных данных пользователю предоставляется доступ на страницу загрузки установочных модулей СКЗИ и комплекта эксплуатационной документации. При загрузке пользователем установочных модулей СКЗИ и комплекта эксплуатационной документации Уполномоченной организацией присваивается учетный номер, идентифицирующий экземпляр СКЗИ, предоставленный пользователю.

На странице загрузки вместе с дистрибутивом и документацией размещаются соответствующие контрольные суммы. Пользователь, загрузив установочные модули СКЗИ и эксплуатационную документацию, должен убедиться в целостности полученных данных с помощью утилиты `cpverify`, полученной доверенным способом, или иного сертифицированного ФСБ России средства.

Установка СКЗИ на рабочее место пользователя может быть осуществлена только в случае подтверждения целостности полученных установочных модулей СКЗИ и эксплуатационной документации.

Примечание.
1) Средство контроля целостности (`cpverify`) первоначально должно быть получено пользователем на физическом носителе в офисе компании ООО «КРИПТО-ПРО», либо у официального дилера. Такая утилита считается полученной доверенным образом. Далее полученной доверенным образом признается очередная версия утилиты, полученная любым образом, например, скачанная с сайта `https://www.cryptopro.ru/`, при условии, что она была проверена другим экземпляром утилиты, полученным ранее доверенным образом, и проверка была успешной.
2) Контроль целостности дистрибутива СКЗИ и компонентов среды функционирования (СФ) СКЗИ обеспечивается при помощи утилиты `cpverify` в соответствии с Приложением 1.

3 Ключевая система и ключевые носители

СКЗИ КриптоПро JCP версии 2.0 R4 является системой с открытым распределением ключей на основе асимметричной криптографии с использованием закрытого ключа на одной стороне и соответствующего ему открытого ключа на другой стороне. Такие пары ключей могут быть двух типов: ключи электронной подписи (ЭП) и ключи обмена.

Ключ ЭП может быть использован только для создания ЭП. Ключ обмена может быть использован как для формирования ключа связи с другим пользователем, так и для создания ЭП.

Пользователь включается в систему и исключается из неё установленным Удостоверяющим Центром порядком.

Пользователь, обладающий правом подписи и/или шифрования данных, вырабатывает на своем рабочем месте или получает у администратора безопасности (в зависимости от принятой политики безопасности) личные закрытый ключ (ключ ЭП) и открытый ключ (ключ проверки ЭП). На основе каждого открытого ключа (ключа проверки ЭП) Удостоверяющим Центром формируется сертификат открытого ключа (сертификат ключа проверки ЭП).

3.1 Шифрование данных

При зашифровании сообщения пользователем А для пользователя Б, пользователь А формирует симметричный ключ связи (сеансовый ключ информационного обмена) на основе своего закрытого ключа обмена и открытого ключа обмена пользователя Б. Соответственно, для расшифрования этого сообщения пользователем Б формируется тот же симметричный ключ на основе своего закрытого ключа обмена и открытого ключа обмена пользователя А.

Таким образом, для обмена данными каждому пользователю необходимо иметь:

- личный закрытый ключ обмена;
- открытые ключи обмена других пользователей.

Сеансовый ключ информационного обмена вырабатывается на основе схемы Диффи-Хеллмана. Схема Диффи-Хеллмана обеспечивает формирование сеансовых ключей, но не обеспечивает аутентификацию связывающихся сторон, поэтому данная схема должна использоваться совместно с протоколами аутентификации.

3.2 Создание и проверка ЭП

Для создания электронной подписи используется ключ электронной подписи, для проверки – соответствующий ключ проверки электронной подписи. Проверяющий должен быть полностью уверен в принадлежности ключа проверки ЭП конкретному пользователю. Для этой цели используется сертификат ключа проверки ЭП, подписанный Удостоверяющим Центром.

Каждому пользователю, обладающему правом подписи, необходимо иметь:

- ключ электронной подписи;
- ключи проверки электронной подписи (сертификаты ключей проверки электронной подписи) других пользователей.

3.3 Ключевые носители

Ключи пользователей могут храниться на различных устройствах, называемых «ключевые носители» («ключевые хранилища»). Ключи на ключевых носителях хранятся в виде специальной структуры, называемой «ключевой контейнер».

Перечень поддерживаемых ключевых носителей в зависимости от программно-аппаратной платформы отражен в ЖТЯИ.00091-04 30 01. КриптоПро JCP. Формуляр, п. 3.10.

Использование носителей других типов допускается только по согласованию с ФСБ России.

Примечание. В состав дистрибутива СКЗИ входят библиотеки для интеграции ключевых носителей в КриптоПро JCP, но не входят модули поддержки и драйвера для ОС. По вопросам получения модулей поддержки и драйверов необходимо обращаться к производителям соответствующих устройств.

При использовании пассивных ключевых носителей необходимо обеспечить выполнение следующих условий:

- подключение съемных носителей должно осуществляться непосредственно к считывателю (считавателю смарт-карт, USB-портам и т.п.), с обеспечением отсутствия канала связи между носителем и СКЗИ, в котором может действовать нарушитель;
- при конструктивном исполнении считывателя ключевого носителя с кабелем необходимо обеспечить нахождение считывателя и кабеля на той же контролируемой территории, что и ПЭВМ.

При невозможности выполнения указанных условий необходимо с учетом модели возможных угроз и нарушителя разработать организационно-технические мероприятия по защите взаимодействия носителя с СКЗИ с последующей оценкой таких мероприятий в рамках проведения соответствующих исследований.

При использовании любых типов ключевых носителей помимо требований эксплуатационной документации на СКЗИ КриптоПро JCP необходимо руководствоваться эксплуатационной документацией на сами носители.

Доступ к ключевому носителю любого типа должен быть защищен паролем и/или PIN-кодом (за исключением случаев, отдельно оговоренных в документации на СКЗИ и используемые носители). Носители могут поддерживать код доступа PUK, используемый для разблокировки носителя. Перед началом работы с ключевым носителем необходимо сменить установленные по умолчанию значения PIN и PUK (при поддержке PIN и PUK на носителе).

3.4 Ключевые контейнеры

Закрытые ключи СКЗИ КриптоПро JCP хранятся в ключевом контейнере, который может содержать в себе:

- только ключ ЭП;
- только ключ обмена;
- ключ ЭП и ключ обмена одновременно.

Единственный ключ ключевого контейнера (ключ ЭП или ключ обмена) называется первичным ключом. Если в контейнере два ключа, то первый ключ (ключ ЭП) называется первичным, второй ключ (ключ обмена) — вторичным.

В СКЗИ КриптоПро JCP при создании ключа, если он уже существует в контейнере с тем же именем (alias), будет получена ошибка, иначе он будет добавлен в контейнер.

Если ключевой контейнер был создан не с помощью КриптоПро JCP (например, при помощи КриптоПро CSP), то в качестве ключа для создания электронной подписи будет использоваться:

- ключ ЭП, если контейнер содержит ключ подписи;
- ключ обмена, если контейнер не содержит ключа подписи.

Кроме ключей, ключевой контейнер содержит служебную информацию, необходимую для обеспечения криптографической защиты ключей, их целостности и т.п.

Каждый ключевой контейнер (независимо от типа носителя) является самодостаточным и содержит всю необходимую информацию для работы как с самим контейнером, так и с закрытыми (и соответствующими им открытыми) ключами.

Ключевой контейнер содержит следующую информацию:

- первичный ключ;
- маски первичного ключа;
- контрольную информацию первичного ключа;
- вторичный ключ (опциональный).

Каждый закрытый ключ хранится в формате, дополнительно содержащем все константы, необходимые для

формирования и экспорта открытого ключа.

Формат ключевого контейнера обеспечивает чтение ключей и соответствующих масок отдельными операциями в раздельные (разнесенные по адресам) области памяти, для чего он содержит шесть зон (реализация зон зависит от типа ключевого носителя).

Ключевой контейнер содержит также дополнительную информацию, необходимую для обеспечения его восстановления при возникновении различных программно-аппаратных сбоев (дополнительная информация включается в тех случаях, когда размер ключевого контейнера не ограничен размерами памяти физического носителя).

3.5 Сертификаты ключа

Сертификат открытого ключа обмена и сертификат ключа проверки ЭП представляет собой структурированную двоичную запись в формате ASN.1, содержащую:

- имя субъекта или объекта системы, однозначно идентифицирующее его в системе;
- открытый ключ или ключ проверки ЭП субъекта или объекта системы;
- дополнительные атрибуты, определяемые требованиями использования сертификата в системе;
- ЭП Издателя (Удостоверяющего центра), заверяющую совокупность этих данных.

Формат сертификата определен в рекомендациях ITU-T X.509, IETF RFC 5280 и рекомендациях ТК26 «Р 1323565.1.023-2018. Использование алгоритмов ГОСТ Р 34.10-2012, ГОСТ Р 34.11-2012 в сертификате, списке аннулированных сертификатов (CRL) и запросе на сертификат PKCS#10 инфраструктуры открытых ключей X.509». В настоящее время основным принятым форматом является формат версии 3, позволяющий определить дополнения (extensions), с помощью которых реализуется определенная политика безопасности в системе.

3.6 Размеры и сроки действия ключей

Размеры ключей электронной подписи:

ключ электронной подписи	256 бит или 512 бит;
ключ проверки электронной подписи	512 бит или 1024 бита.

Размеры ключей, используемых при шифровании:

закрытый ключ	256 бит или 512 бит;
открытый ключ	512 бит или 1024 бита;
симметричный ключ	256 бит.

В случае использования Java-машин версии 1.7 или 1.8:

Возможна ситуация, когда установленная JRE имеет экспортные ограничения. США запрещает экспорт "сильной" криптографии и КриптоПро JCP с длинами ключа 256 или 512 бит попадает под это ограничение. Ограничения устанавливаются файлами local_policy.jar и US_export_policy.jar в каталоге <JRE>/jre/lib/security. Для снятия экспортных ограничений необходимо скачать файл jce_policy.zip с политиками со страницы http://www.oracle.com/technetwork/java/javase/downloads/index.html, выбирая "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files" версии 7 или 8. Для отладки же можно просто скопировать US_export_policy.jar в local_policy.jar (оба файла должны присутствовать).

При эксплуатации СКЗИ КриптоПро JCP версии 2.0 R4 должны соблюдаться следующие сроки использования пользовательских закрытых ключей и сертификатов:

- максимальный срок действия ключа ЭП — 1 год 3 месяца;
- максимальный срок действия ключа проверки ЭП — 15 лет после окончания срока действия соответствующего закрытого ключа;
- максимальный срок действия сертификата ключа проверки ЭП — 5 лет;

- максимальный срок действия закрытых и открытых ключей обмена — 1 год 3 месяца;

Срок действия ключа берется из (в порядке уменьшения приоритета):

- Расширения контейнера ключа;
- Расширения сертификата ключа;
- Даты создания ключа + 1 год 3 месяца.

При формировании закрытого ключа в контейнер записывается дата истечения срока действия этого ключа. В зависимости от включения параметра «Проверять срок действия закрытого ключа» (см. рис. 2) возможно ограничение использование ключа по истечении его срока действия.

Примечание. При работе в режиме усиленного контроля использования ключей (режим обязателен к использованию, отключение может производиться только в целях тестирования) режим «Проверять срок действия закрытого ключа» по умолчанию включен.

3.6.1 Усиленный контроль использования ключей

Режим усиленного контроля использования ключей обеспечивает осуществление контроля срока действия долговременных ключей электронной подписи и ключевого обмена, контроля доверенности ключей проверки электронной подписи и контроля корректного использования программного датчика случайных чисел.

Данный режим должен быть в обязательном порядке включён при инсталляции СКЗИ (см. рис. 1), либо через контрольную панель КриптоПро JCP версии 2.0 R4 (вкладка «Дополнительно») перед началом использования СКЗИ (см. рис. 2).

Примечание. Работа СКЗИ при отключённом режиме усиленного контроля использования ключей допускается исключительно в тестовых целях.

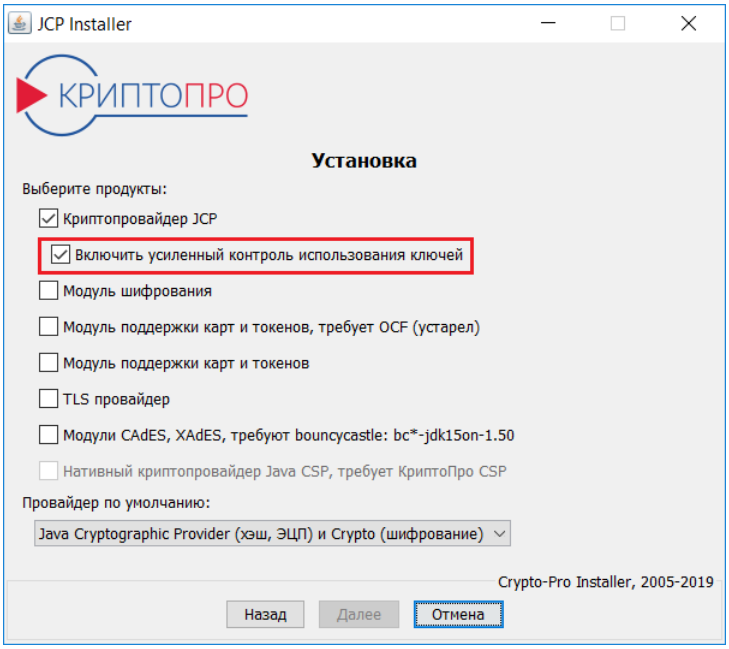

Рисунок 1. Включение режима усиленного контроля использования ключей при инсталляции СКЗИ

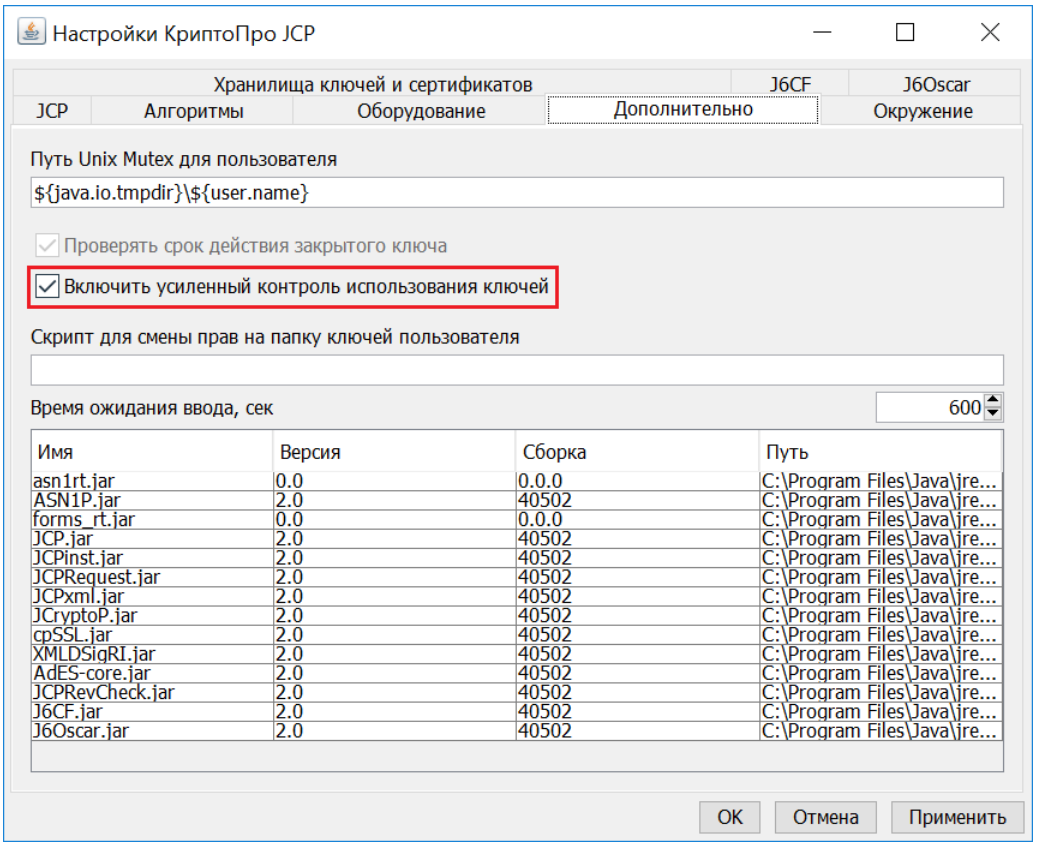

Рисунок 2. Включение режима усиленного контроля использования ключей

3.7 Управление ключами

Ключевая система СКЗИ базируется на архитектуре PKI рекомендаций X.509. Формирование и управление сертификатами открытых ключей производится УЦ. В качестве УЦ может выступать Удостоверяющий центр КриптоПро УЦ или другие сертифицированные ФСБ России УЦ, обеспечивающие выполнение функций доверенного обращения с сертификатами. Также допускается использование Центра Сертификации корпорации Microsoft (Microsoft Certification Authority) в тестовых целях.

СКЗИ КриптоПро JCP версии 2.0 R4 может использоваться в качестве криптоядра в составе различных прикладных систем, организационные схемы управления ключевой системой которых могут отличаться от рассматриваемой.

Управление ключами может осуществляться при помощи программы keytool, входящей в состав Java Runtime, или при помощи прикладного программного обеспечения (см. ЖТЯИ.00091-04 33 01. КриптоПро JCP. Руководство программиста, раздел «Использование утилиты keytool»).

Подробные рекомендации по управлению ключевой системой на всех этапах ее жизненного цикла приведены в эксплуатационной документации на КриптоПро УЦ. Взаимодействие с компонентами УЦ при управлении ключами должно осуществляться в соответствии с Регламентом УЦ.

3.7.1 Формирование ключей

Формирование ключей пользователя производится через стандартный интерфейс JCA при помощи класса `KeyPairGenerator` (см. ЖТЯИ.00091-04 33 01. КриптоПро JCP. Руководство программиста)

Примечание.
1) Ключи ЭП и обмена формируются с использованием программного ДСЧ с инициализацией от биологического ДСЧ (клавиатура-мышь), обеспечивающих защиту по уровню КС1.
2) При использовании считывателей смарт-карт необходимо произвести настройки OpenCard Framework (см. п. 1.8 ЖТЯИ.00091-04 91 01. КриптоПро JCP. Инструкция по использованию).
3) Перед использованием процессорные карты должны быть "выпущены" с использованием транспортного пин-кода и ПО выпуска карт (поставляются дистрибутором карт).
4) При использовании НГМД в качестве ключевого носителя во избежание потери ключевой информации рекомендуется хранить ее копию.

В связи с переходом на использование алгоритма ГОСТ Р 34.10-2012 и соответствующем запрете использования алгоритма ГОСТ Р 34.10-2001, при попытке генерации или создания подписи с использованием ключа алгоритма ГОСТ Р 34.10-2001 будет выдано следующее предупреждение (см. рис. 3).

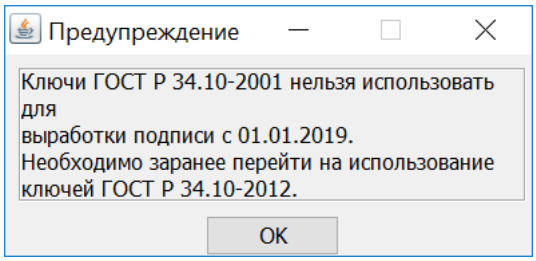

Рисунок 3. Предупреждение о запрете использования ключей алгоритма ГОСТ Р 34.10-2001

Создание подписи с использованием ключа алгоритма ГОСТ Р 34.10-2001 с 01 января 2020 года запрещено.

3.7.2 Хранение ключевых носителей

При хранении ключей и ключевых носителей должны выполняться следующие требования:

1) Необходимо обеспечить невозможность доступа к ключевым носителям не допущенных к ним лиц. Личные ключевые носители пользователей рекомендуется хранить в сейфе. Пользователь несет персональную ответственность за хранение личных ключевых носителей.
2) В случае централизованного хранения ключевых носителей в организации, эксплуатирующей СКЗИ, администратор безопасности (если он имеется) несет персональную ответственность за хранение личных ключевых носителей пользователей.
3) Запрещается оставлять без контроля вычислительные средства с установленным СКЗИ после ввода ключевой информации.
4) Хранение ключей на несъемных носителях допускается только при условии распространения на носитель требований по обращению с ключевыми носителями (в том числе и после удаления ключей).
5) В случае невозможности отчуждения ключевого носителя с ключевой информацией от ПЭВМ организационно-техническими мероприятиями должен быть исключен доступ нарушителей к ПЭВМ с ключами.
6) При хранении ключей рекомендуется использовать парольную защиту.
7) При необходимости передачи ключевого носителя постороннему лицу информацию с носителя необходимо гарантированно удалить.
8) В случае необходимости проведения ремонтных и регламентных работ аппаратной части СФ необходимо обеспечить невозможность доступа нарушителя к ключевой информации, содержащейся в СФ. Конкретный перечень мер должен быть определен исходя из условий эксплуатации СКЗИ.

3.7.3 Компрометация ключей пользователя

К событиям, связанным с компрометацией ключей относятся, в частности, следующие:

1) потеря ключевых носителей;
2) потеря ключевых носителей с их последующим обнаружением;

3) увольнение сотрудников, имевших доступ к ключевой информации;
4) нарушение правил хранения и уничтожения (после окончания срока действия) закрытого ключа;
5) возникновение подозрений на утечку информации или ее искажение в системе конфиденциальной связи;
6) нарушение печати на сейфе с ключевыми носителями;
7) случаи, когда нельзя достоверно установить, что произошло с ключевыми носителями (в том числе случаи, когда ключевой носитель вышел из строя и доказательно не опровергнута возможность того, что, данный факт произошел в результате несанкционированных действий злоумышленника).

Различаются два вида компрометации закрытого ключа: явная и неявная. Первые четыре события трактуются как явная компрометация ключей. Следующие три требуют специального рассмотрения в каждом конкретном случае.

При компрометации своего ключа пользователь должен немедленно прекратить связь по сети с другими пользователями. При этом пользователь (или администратор безопасности организации) должен немедленно известить ЦР (УЦ) о компрометации ключа пользователя.

После компрометации ключей пользователь формирует новый закрытый ключ и запрос на сертификат. Так как пользователь не может использовать скомпрометированный ключ для формирования ЭП и передачи запроса в защищенном виде по сети, запрос на сертификат вместе с бланками доставляется лично пользователем (администратором безопасности) в Центр Регистрации.

3.7.4 Уничтожение ключей на ключевых носителях

Ключи на ключевых носителях (включая смарт-карты), срок действия которых истек, уничтожаются путем удаления ключевых контейнеров средствами ПО СКЗИ или с помощью «Контрольной Панели», после чего ключевые носители могут использоваться для записи на них новой ключевой информации.

4 Требования по встраиванию и использованию ПО СКЗИ

Для обеспечения защиты электронных документов и создания защищенной автоматизированной системы в первую очередь используют криптографические методы защиты, которые позволяют обеспечить защиту целостности, авторства и конфиденциальности электронной информации и реализовать их в виде программных или аппаратных средств, встраиваемых в автоматизированную систему.

Использование криптографических средств требует, как правило, применения также организационно-технических мер защиты.

Следует отметить, что вновь разрабатываемые СФ и прикладное ПО, использующие сертифицированные СКЗИ должны удовлетворять требованиям к СКЗИ в части корректности использования СКЗИ и проверки влияния на СКЗИ со стороны ПО. Встраивание СКЗИ могут производить организации, имеющие лицензию на право проведения таких работ, а работы по встраиванию должны проводиться в соответствии с Положением ПКЗ 2005.

При создании защищенной автоматизированной системы необходимо определить модель угроз и политику безопасности. В зависимости от политики безопасности определяется необходимый набор криптографических функций и организационно-технических мер, реализуемых в создаваемой системе.

СКЗИ КриптоПро JCP версии 2.0 R4 в первую очередь предназначено для встраивания в прикладное программное обеспечение. Функции СКЗИ могут быть использованы:

- через интерфейс функций JCA, что позволяет применять весь инструментарий Java. Для этих целей разработчики могут воспользоваться программной документацией, содержащейся в Java SDK, а также поставляемым тестовым ПО. Для этих целей в комплект поставки включается документ ЖТЯИ.00091-04 33 01. КриптоПро JCP. Руководство программиста.
- в стандартном прикладном ПО Java.

Встраивание СКЗИ КриптоПро JCP версии 2.0 R4 производится с учетом положений п. 1.5 Формуляра ЖТЯИ.00091-04 30 01.

Исследования СФ не требуются, если в качестве СФ используется веб-сервер в составе ПО Apache Tomcat (Tomcat 7.0, Tomcat 8.5, Tomcat 9.0) в соответствии с требованиями п. 7.4 ЖТЯИ.00091-04 33 03. Руководство программиста (JTLS).

4.1 Правила встраивания и использования СКЗИ

При встраивании СКЗИ КриптоПро JCP версии 2.0 R4 в прикладное программное обеспечение или использовании его в составе стандартного прикладного ПО должны выполняться следующие требования:

1) При использовании ключей проверки ЭП должна быть обеспечена целостность и идентичность ключа проверки ЭП.

2) При использовании сертификатов ключей проверки подписи, заверенных подписью доверенной стороны, должна быть обеспечена безопасная доставка и хранение сертификата ключа ЭП доверенной стороны, с использованием которого проверяются остальные сертификаты ключей проверки пользователей.

3) Криптографическое средство, с помощью которого производится заверение ключей проверки ЭП или справочников ключей проверки ЭП, должно быть сертифицировано по классу, соответствующему принятой политике безопасности.

4) Для отзыва (вывода из действия) ключей проверки подписи должны использоваться средства, позволяющие произвести авторизацию отзывающего лица (в этих целях может быть использован список отозванных сертификатов, заверенный ЭП доверенной стороны).

5) При вызове функций СКЗИ КриптоПро JCP версии 2.0 R4 в прикладном программном обеспечении необходимо, при возникновении критических исключений блокировать криптографические вызовы, а при возникновении других исключений, корректно их обрабатывать (подробнее см. ЖТЯИ.00091-04 33 01. КриптоПро JCP. Руководство программиста).

5 Требования по проведению контроля целостности

5.1 Контроль целостности JAR-файлов провайдера

При загрузке провайдера КриптоПро JCP осуществляется проверка подписей трех файлов jcp.jar, ASN1P.jar и asn1rt.jar на заданном ГОСТ сертификате (сертификат «прошит» в классе `JarChecker`), а также выполняется проверка DSA-подписи на сертификате Oracle.

Контроль целостности осуществляется при помощи функции `check()` класса `JarChecker`, которая запускается в классе `Starter` загружаемого провайдера, собранного с отключенным флагом DEBUG. Для корректной работы данной функции сертификат прошит в самом классе `JarChecker`, и указание пути к файлу с сертификатом при осуществлении контроля целостности не требуется.

Функции создания и проверки подписи JAR-файлов класса `JarChecker` описаны в Приложении 1.

5.2 Командная строка CPVerify

Командная строка CPVerify введена для более удобного контроля целостности, а также возможности администрирования систем, в которых отсутствует графическое расширение или пакеты Java AWT и Swing.

С помощью командной строки можно создавать хранилища хэшей, добавлять в них файлы, удалять файлы из хранилища, пересчитывать и проверять хэши файлов в хранилище, а также работать с основным системным хранилищем.

Общий синтаксис вызова командной строки описан в Приложении 1.

5.3 Список файлов, добавляемых в хранилище для контроля целостности

Перед началом работы с провайдером необходимо создать основное хранилище в системных настройках, права на изменения которого даны только администратору. В него следует добавить исполняемые файлы Java, файлы, содержащие пакеты java.lang, java.security, java.io, javax, системные библиотеки, конфигурационные файлы Java-машины, модули КриптоПро JCP, которые находятся в каталоге `[P»CЋCЂCÆPəJRE]/lib/ext`.

Под контролем целостности должны находится следующие .jar файлы:

AdES-core.jar
ASN1P.jar
asn1rt.jar
CAdES.jar
cpSSL.jar
forms_rt.jar
J6CF.jar
J6Oscar.jar
JCP.jar
JCPControlPane.jar
JCPinst.jar
JCPinstGUI.jar
JCPRequest.jar
JCPRevCheck.jar
JCPRevTools.jar
JCPxml.jar
JCryptoP.jar
Rutoken.jar
XAdES.jar
XMLDSigRI.jar

Этот список следует проверять средствами командной строки в отдельном процессе до загрузки рабочей Java-машины не реже одного раза в сутки.

В случае, если `cpverify` зафиксирует изменение хотя бы одного элемента Java-машины, использование криптопровайдера следует прекратить до выяснения и устранения причин возникновения ошибки.

6 Требования по защите от НСД

6.1 Общие требования по организации работ по защите от НСД

Защита аппаратного и программного обеспечения от НСД при установке и использовании СКЗИ КриптоПро JCP версии 2.0 R4 является составной частью общей задачи обеспечения безопасности информации в системе, в состав которой входит СКЗИ.

Защита информации от НСД в автоматизированной системе обеспечивается комплексом программно-технических средств и поддерживающих их организационных мер. В их число входит:

- применение специальных программно-аппаратных средств защиты;
- организация системы контроля безопасности информации;
- физическая охрана ПЭВМ и ее средств;
- администрирование информационной безопасности;
- учет носителей информации;
- сигнализация о попытках нарушения защиты;
- периодическое тестирование технических и программных средств защиты;
- использование сертифицированных программных и технических и средств.

Наряду с применением средств защиты от НСД необходимо выполнение приведенных ниже организационно-технических и административных мер по обеспечению правильного функционирования средств обработки и передачи информации, а также установление соответствующих правил для обслуживающего персонала, допущенного к работе с СКЗИ.

Защита информации от НСД должна обеспечиваться на всех технологических этапах обработки информации и во всех режимах функционирования, в том числе при проведении ремонтных и регламентных работ.

В организации, эксплуатирующей СКЗИ, должен быть назначен администратор безопасности, на которого возлагаются задачи организации работ по использованию СКЗИ, выработки соответствующих инструкций для пользователей, а также контроль за соблюдением требований по безопасности.

Администратор безопасности не должен иметь возможности доступа к конфиденциальной информации пользователей.

Правом доступа к рабочим местам с установленными СКЗИ должны обладать только определенные для эксплуатации лица, прошедшие соответствующую подготовку. Администратор безопасности должен ознакомить каждого пользователя, применяющего СКЗИ, с документацией на СКЗИ, а также с другими нормативными документами, созданными на её основе.

Защита информации от НСД должна предусматривать контроль эффективности средств защиты от НСД. Этот контроль должен периодически выполняться администратором безопасности на основе требований документации на средства защиты от НСД.

6.2 Требования по размещению технических средств с установленным СКЗИ

При осуществлении доступа в глобальные сети передачи данных непосредственно с рабочих мест, оснащенных СКЗИ КриптоПро JCP версии 2.0 R4, должны быть приняты меры, исключающие возможность воздействия нарушителя на СКЗИ по каналам связи, выходящим за пределы контролируемой зоны.

В целях защиты открытой конфиденциальной информации от утечки по техническим каналам, в том числе по каналам связи, от объектов информатизации и СКЗИ, ввод в действие и эксплуатация указанных объектов и СКЗИ должна осуществляться в соответствии с действующими в Российской Федерации требованиями по защите информации от утечки по техническим каналам, в том числе по каналу связи (например, СТР-К).

Необходимость и достаточность мер, в том числе по каналу связи, должна оцениваться порядком, предусмотренным упомянутыми руководящими документами, с учетом целевых установок предполагаемого нарушителя

и угроз безопасности информации, определяемых моделью угроз и нарушителя. При этом, если объекты аттестованы на соответствие установленным требованиям по защите информации без учета оценки канала связи, при подключении таких средств к каналам связи, выходящим за пределы контролируемой территории, необходимо использовать любое из следующих средств:

- Волоконно-оптические линии связи;
- Оптические развязывающие устройства, устанавливаемые в тракт передачи информации для создания оптоволоконного фрагмента сети;
- Сертифицированные СКЗИ для передачи информации соответствующего уровня конфиденциальности.

Для обеспечения защиты информации по классу КС от утечки по каналу линейной передачи достаточно, чтобы канал связи был реализован в виде радиоканала GSM, либо GPRS, либо 3G/4G, либо Wi-Fi, либо другого канала мобильной и беспроводной связи, работающего в диапазоне частот несущей выше 800 МГц с цифровой модуляцией штатного информационного сигнала.

6.3 Меры по обеспечению защиты от НСД

При использовании СКЗИ КриптоПро JCP версии 2.0 R4 необходимо наличие механизма локальной аутентификации пользователей ОС.

Необходимо разработать и применить политику назначения и смены паролей в соответствии со следующими правилами:

1) Длина пароля должна быть не менее 8 символов;

2) Количество подряд следующих попыток аутентификации одного субъекта доступа должно быть не более 10. При превышении числа подряд идущих попыток аутентификации одного субъекта доступа установленного предельного значения доступ этого субъекта к СКЗИ блокируется на сутки;

3) Недопустимо при выборе каждого символа пароля ограничиваться менее, чем 10 вариантами;

4) Периодичность смены пароля не должна превышать 6 месяцев;

5) Для обеспечения требований к установлению пароля должны выполняться следующие настройки операционных систем:

Linux:

В файле `/etc/login.defs` параметр `LOGIN_RETRIES` не должен превышать 10, `PASS_MAX_DAYS=180`.

В файле `etc/pam.d/common-password` необходимо добавить опцию min:

```
password      [success=1 default=ignore]      pam_unix.so obscure sha512 min=10
```

AIX:

В файле `/etc/security/user` в секции `default` необходимо установить следующие значения:

```
minlen = 6
maxrepeats = 10
maxage=24
```

Solaris:

В файле `/etc/default/passwd` необходимо установить следующие значения:

```
PASSLENGTH=6
MAXREPEATS=10
MAXWEEKS=24
```

Windows:

В групповых политиках перейти в Local Computer Policy⇒Computer Configuretion⇒Windows Settings⇒Security Settings⇒Account Policies, установить следующие параметры:

```
Minimum password length = 6
Maximum password age = 24
```

Перейти в Local Computer Policy⇒Computer Configuretion⇒Windows Settings⇒Security Settings⇒Account Lockout Policie, установить параметр:

```
Account Lockout threshold = 10
```

При использовании СКЗИ должны выполняться следующие меры по защите информации от НСД:

1) Провайдер КриптоПро JCP версии 2.0 R4 должен использоваться в среде, защищенной от действий внешнего нарушителя, и в корпоративных сетях, защищенных от внутреннего нарушителя.

2) Необходимо ограничить возможность вывода информации с используемой ПЭВМ через порты COM, LPT, USB, IEEE 1394, а также средствами Bluetooth, Wi-Fi и аналогичных.

3) Право доступа к рабочим местам с установленным ПО СКЗИ предоставляется только лицам, ознакомленным с правилами пользования и изучившим эксплуатационную документацию на программное обеспечение, имеющее в своем составе СКЗИ.

4) Необходимо запретить осуществление не санкционированного администратором безопасности копирования ключевых носителей.

5) Необходимо запретить разглашение содержимого ключевых носителей и передачу самих носителей лицам, к ним не допущенным, а также выводить ключевую информацию на дисплей и принтер.

6) Необходимо запретить использование ключевых носителей в режимах, не предусмотренных правилами пользования СКЗИ, либо использовать ключевые носители на посторонних ПЭВМ.

7) Необходимо запретить запись на ключевые носители посторонней информации.

8) Требования по хранению личных ключевых носителей распространяются на ПЭВМ (в том числе и после удаления ключей с диска).

9) На технических средствах, оснащенных СКЗИ, должно использоваться только лицензионное программное обеспечение фирм-производителей.

10) На ПЭВМ, оснащенных СКЗИ, не допускается установка средств разработки и отладки ПО. Если средства отладки приложений необходимы для технологических потребностей пользователя, то их использование должно быть санкционировано администратором безопасности. В любом случае запрещается использовать эти средства для просмотра и редактирования кода и памяти приложений, использующих СКЗИ.

11) Необходимо исключить попадание в систему программ, позволяющих, пользуясь ошибками ОС, получать привилегии администратора.

12) Должны быть приняты меры по исключению несанкционированного доступа посторонних лиц в помещения, в которых установлены технические средства СКЗИ, по роду своей деятельности не являющихся персоналом, допущенным к работе в указанных помещениях.

13) Должно быть запрещено оставлять без контроля вычислительные средства, на которых эксплуатируется СКЗИ, после ввода ключевой информации. При уходе пользователя с рабочего места должно использоваться автоматическое включение парольной заставки.

14) Администратором безопасности должно быть проведено опечатывание системного блока с установленным СКЗИ, исключающее возможность несанкционированного изменения аппаратной части рабочей станции.

15) Из состава системы должно быть исключено все оборудование, которое может создавать угрозу безопасности ОС. Также избегают использования любых нестандартных аппаратных средств, имеющих возможность влиять на нормальный ход работы компьютера или ОС.

16) При использовании СКЗИ на ПЭВМ, подключенных к общедоступным сетям связи, должны быть предприняты дополнительные меры, исключающие возможность несанкционированного доступа к системным ресурсам используемых операционных систем, к программному обеспечению, в окружении которого функционируют СКЗИ, и к компонентам СКЗИ со стороны указанных сетей.

17) Должны быть отключены радиоканалы, неиспользуемые в работе СКЗИ.

18) Необходимо регулярно устанавливать пакеты обновления безопасности ОС (Service Packs, Hot fix и т.п.), обновлять антивирусные базы, а также исследовать информационные ресурсы по вопросам компьютерной безопасности с целью своевременной минимизации опасных последствий от возможного воздействия на ОС.

19) Правом установки и настройки ОС и СКЗИ должен обладать только администратор безопасности.

20) В BIOS ПЭВМ определяются установки, исключающие возможность загрузки операционной системы, отличной от установленной на жестком диске: отключается возможность загрузки с гибкого диска, привода CD-ROM и прочие нестандартные виды загрузки ОС, включая сетевую загрузку. Не применяются ПЭВМ с BIOS, исключающим возможность отключения сетевой загрузки ОС.

21) Средствами BIOS должна быть исключена возможность отключения пользователями ISA и PCI устройств при использовании ПАК защиты от НСД, устанавливаемых в ISA и PCI разъем.

22) Вход в BIOS ПЭВМ должен быть защищен паролем, к которому предъявляются те же требования, что и к паролю администратора. Пароль для входа в BIOS должен быть известен только администратору и быть отличным от пароля администратора для входа в ОС.

23) Средствами BIOS должна быть исключена возможность работы на ПЭВМ, если во время его начальной загрузки не проходят встроенные тесты.

24) При загрузке ОС должен быть реализован контроль целостности программного обеспечения, входящего в состав СКЗИ, самой ОС и всех исполняемых файлов, функционирующих совместно с СКЗИ.

25) Должно быть реализовано физическое затирание содержимого удаляемых файлов, в том числе SWAP файла.

26) Должно быть обеспечено тестирование аппаратуры в объеме самотестирования при перезагрузке не реже чем 1 раз в 15 суток.

27) Переставлять реализацию класса Preferences с помощью property java.util.prefs.PreferencesFactory запрещается.

28) Необходимо обеспечить административными мерами контроль доступа к системным и пользовательским настройкам Java-машины.

При использовании КриптоПро JCP на платформе Windows системные и пользовательские настройки Java-машины хранятся в реестре в разделах `HKEY_CURRENT_USER\Software\JavaSoft\Prefs` и `HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Prefs` соответственно.

На платформах Solaris и Linux системные и пользовательские настройки Java-машины хранятся в файловой системе. Системные настройки находятся в каталоге `.systemPrefs`, положение которого определяется переменной `java.util.prefs.systemRoot` (по умолчанию `/etc/.java`). Если он недоступен, то `.systemPrefs` находится в каталоге, определенном переменной `java.home`. Пользовательские настройки находятся в каталоге .java/.userPrefs, положение которого определяется переменной `java.util.prefs.userRoot`. Если переменная не задана, то каталог `.java/.userPrefs` находится в каталоге, определенном переменной `user.home`.

6.4 Меры обеспечения безопасности функционирования рабочих мест со встроенными СКЗИ

1) Использование шифровальных средств для криптографической защиты информации подлежит лицензированию в соответствии с действующим законодательством РФ.

2) Рабочие места, на которые установлены СКЗИ, должны быть аттестованы комиссией.

3) Правом доступа к рабочим местам с установленным СКЗИ должны обладать только лица, прошедшие соответствующую подготовку. Администратор безопасности должен ознакомить каждого пользователя, использующего СКЗИ, с правилами пользования или с другими нормативными документами, созданными на их основе.

4) Должностные инструкции администратора безопасности (его заместителя) и ответственного исполнителя должны учитывать требования настоящих Правил.

5) При каждом включении рабочей станции с установленным СКЗИ необходимо проверять сохранность печатей системного блока и разъемов рабочей станции.

6) Администратор безопасности должен периодически (не реже 1 раза в 2 месяца) проводить контроль целостности и легальности установленных копий ПО на всех АРМ со встроенной СКЗИ с помощью программ контроля целостности.

7) В случае обнаружения «посторонних» (не зарегистрированных) программ, нарушения целостности программного обеспечения либо выявления факта повреждения печатей на системных блоках работа на АРМ должна быть прекращена. По данному факту должно быть проведено служебное расследование комиссией в составе представителей служб информационной безопасности организации-владельца сети и организации-абонента сети, где произошло нарушение, и организованы работы по анализу и ликвидации негативных последствий данного нарушения.

8) Не допускается оставлять без контроля вычислительные средства, входящие в состав СКЗИ, при включенном питании и загруженном программном обеспечении СКЗИ. При кратковременном перерыве в работе рекомендуется производить гашение экрана, возобновление активности экрана производится с использованием пароля доступа.

9) Пользователь должен запускать только те приложения, которые разрешены администратором безопасности.

10) Средствами BIOS должна быть исключена возможность отключения пользователями ISA-устройств.

11) ПО, установленное на ПЭВМ, не должно иметь встроенных средств разработки и отладки программ.

12) ПЭВМ, обеспечивающие удаленный вход пользователей из глобальной сети, не должны использовать ПО СКЗИ.

13) Пароли, назначаемые пользователям, должны отвечать требованиям соответствующих инструкции и нормативных документов.

При эксплуатации СКЗИ **не допускается**:

1) Осуществлять несанкционированное копирование ключевых носителей.

2) Разглашать содержимое носителей ключевой информации или передавать сами носители лицам, к ним не допущенным, выводить ключевую информацию на дисплей и принтер (за исключением случаев, предусмотренных данными правилами).

3) Вставлять ключевой носитель в устройство считывания в режимах, не предусмотренных штатным режимом использования ключевого носителя.

4) Подключать к ПЭВМ дополнительные устройства и соединители, не предусмотренные штатной комплектацией.

5) Работать на компьютере, если во время его начальной загрузки не проходит встроенный тест ОЗУ, предусмотренный в ПЭВМ.

6) Вносить какие-либо изменения в программное обеспечение СКЗИ.

7) Изменять настройки, установленные программой установки СКЗИ или администратором.

8) Использовать синхропосылки, вырабатываемые не средствами СКЗИ.

9) Обрабатывать на ПЭВМ, оснащенной СКЗИ, информацию, содержащую государственную тайну.

10) Использовать бывшие в работе ключевые носители для записи новой информации без предварительного уничтожения на них ключевой информации средствами СКЗИ.

11) Осуществлять несанкционированное вскрытие системных блоков ПЭВМ.

12) Приносить и использовать в помещении, где размещены средства СКЗИ, радиотелефоны и другую радиопередающую аппаратуру.

Приложение 1
Контроль целостности программного обеспечения

Контроль целостности JAR-файлов

Для **создания подписи JAR-файла** следует вызвать функцию `main` класса `JarChecker` со следующими параметрами:

```
-sign [-alias keyAlias] [-storetype storeType] [-keypass keyPassword]
          [-in jar_file] [-out signed_jar_file] [-delsign]
```

-alias имя ключа электронной подписи, на котором осуществляется подпись

-storetype имя ключевого носителя, на котором лежит ключ, по умолчанию HDImageStore; параметр обязателен, если ключ лежит не на жестком диске

-keypass пароль на ключ подписи (если установлен)

-in полный путь к подписываемому JAR-файлу

-out полный путь к файлу, в который будет записан подписанный JAR-файл

-delsign флаг, определяющий удаление неверных подписей; если указан, то неверные подписи будут удалены

Процесс создания подписи:

1) в первую очередь осуществляется проверка всех существующих подписей;если существуют неверные подписи и указан параметр -delsign, то эти подписи удаляются;

2) если после проверки и, возможно, удаления неверных подписей, общее количество подписей равно и превышает 16, то новая подпись создаваться не будет;

3) если среди оставшихся подписей уже существует подпись на заданном ключе электронной подписи и она верна, то новая подпись не создается;

4) если среди оставшихся подписей уже существует подпись на заданном ключе электронной подписи и она не верна, то неверная подпись удаляется (вне зависимости от флага -delsign) и создается новая подпись на этом ключе; в противном случае просто формируется новая подпись.

В процессе формирования подписи JAR-файла осуществляется копирование всех классов, входящих в исходный JAR-файл, в новый JAR-файл, определенный путем -out. После чего в директории META-INF нового JAR-файла создаются два файла: Digest.CP и Sign.CP (если подпись производится не в первый раз, т.е. если такие файлы уже существуют в исходном JAR-файле, то файл Digest.CP не изменяется, а к содержимому Sign.CP добавляется информация о новой подписи).

Первый файл представляется собой набор пар: имя класса, входящего в JAR-файл, но не входящего в директорию META-INF, и значение хэша на содержимое этого класса. Второй файл содержит в себе следующую информацию:

- количество подписей (уже существующих + только что созданная);
- набор подписей (уже существующие + новая подпись);
- набор соответствующих подписям сертификатов (уже существующие + новый сертификат).

Новая подпись (как и все предыдущие) производится на содержимое файла Digest.CP.

Для **проверки подписи JAR-файла** следует вызвать функцию `main` класса `JarChecker` со следующими параметрами:

```
-verify  [-in signed_jar_file] [-cert cert_file]
```

-in полный путь к подписанному JAR-файлу

-cert полный путь к сертификату, на котором осуществляется проверка подписи; если указан, то осуществляет проверка подписи, соответствующей заданному сертификату, в противном случае осуществляется проверка всех имеющихся подписей JAR-файла

Процесс проверки подписи:

1) если в JAR-файле, подпись которого проверяется, не существует файлов Digest.CP и Sign.CP, то файл не содержит подписей и никакой проверки не требуется;

2) если такие файлы имеются, то в первую очередь осуществляется подсчет хэшей всех классов подписанного JAR-файла, не входящих в директорию META-INF, и проверка соответствия результата значениям хэшей, содержащихся в файле Digest.CP. Если результаты различны, то выдается Exception (считается, что JAR-файл поврежден);

3) если хэши сошлись, то осуществляется собственно проверка подписи:

а) если задан конкретный сертификат (командой -cert), то производится поиск соответствующего сертификата в файле Sign.CP и, если такой сертификат найден, осуществляется проверка соответствующей подписи;

б) в противном случае осуществляется проверка всех подписей, содержащихся в файле Sign.CP. При этом после проверки на экран выводится информация о всех неверных подписях.

Командная строка CPVerify

Командной строке Prompt всегда передается следующий набор параметров:

```
doing repository [params]
```

Параметр **doing** определяет действие, требуемое от командной строки. Он обязательно должен стоять первым. Допустимые значения параметра:

-verify Проверить файлы в хранилище. Команда проверяет хэши одного или нескольких файлов из указанного хранилища.

-make Пересчитать хэши файлов в хранилище. По команде пересчитывается хэш одного или нескольких файлов в хранилище.

-add Добавить файлы в хранилище. Команда добавляет один или несколько файлов в хранилище, и пересчитывает для них хэш.

-delete Удалить файлы из хранилища. Команда удаляет один или несколько файлов из хранилища.

-create Создать хранилище. Команда создает новое хранилище. Если хранилище уже существует, то оно будет перезаписано.

-check Проверить статус хранилища. Команда проверяет статус хранилища: не повреждено или не удалено ли оно.

-setdefault Сделать хранилище основным системным. Основное системное хранилище - то, в котором хранятся хэши наиболее важных системных файлов, и которое периодически проверяют внутренние службы криптопровайдера.

-getdefault Узнать, какое хранилище является основным системным.

-print Вывести состояние всех файлов в хранилище. Команда выводит состояние всех файлов в хранилище: не повреждены ли они, не удалены ли.

-help Вывести справку. Общая справка по всем командам.

Параметр **repository** является необходимым во всех командах, кроме **-help** и **-getdefault**. Он задает хранилище, с которым будет проводиться операция. Он может быть определен одним из трех следующих способов:

-repfile filename Хранилище — файл, с именем filename.

-reppref Хранилище расположено в каталоге системных настроек (реестр для Windows).

-repdefaul Основное системное хранилище.

Примечание. Параметр **repository** может быть любым по счету, но не первым. Если используется значение **-repfile**, следующее слово в командной строке считается именем файла.

Параметры **[params]** зависят от команды. Для всех команд действует параметр **-help** — подробная справка по команде. Команды **-help, -print, -getdefault, -setdefault, -check, -create** не предполагают дополнительных параметров. В командах **-verify, -make, -add, -delete** надо определять список файлов, над которыми производится действие, специальными параметрами, указанными ниже:

-verify	-all	Проверить все файлы, лежащие в хранилище.
	-file filename1 [-file filename2 [...]]	Проверить файлы filename1, filename2,..., если они есть в данном хранилище.
-make	-all	Пересчитать хэши всех файлов, лежащих в хранилище.
	-file filename1 [-file filename2 [...]]	Пересчитать хэши файлов filename1, filename2,..., лежащих в хранилище.
-add	-file filename1 [-file filename2 [...]]	Добавить в хранилище файлы filename1, filename2,...
-delete	-all	Удалить все файлы из хранилища.
	-file filename1 [-file filename2 [...]]	Удалить файлы filename1, filename2,... из хранилища.

Любая команда из тех, которые изменяют хранилище, перед сохранением хранилища вызывает проверку всех файлов, в нем содержащихся. Поэтому если какие-то файлы в хранилище повреждены, его состояние можно перезаписать только удалив из него все файлы одной командой -delete, или пересчитав для всех файлов хэш.

Приложение завершается с ошибкой (выход по исключению), если возникла непредвиденная ситуация (отказано в доступе к файлу, ошибка ввода-вывода и т.д.), или если возникла ошибка при работе с хранилищем, когда оно считается существующим. Так, если проверяется статус поврежденного хранилища, приложение завершится нормально, выдав диагностическое сообщение, однако если проводится проверка файлов в поврежденном хранилище, приложение завершится с ошибкой. Приложение завершается нормально, если проверяемый файл из хранилища поврежден, выдавая соответствующую диагностику.

Приложение 2
Перечень вызовов, использование которых при разработке систем на основе СКЗИ КриптоПро JCP с учетом п.1.8 Формуляра ЖТЯИ.00091-04 30 01 возможно без дополнительных тематических исследований

Разработка программного обеспечения на основе СКЗИ КриптоПро JCP версии 2.0 R4 может производиться без создания новых СКЗИ в случае использования вызовов из перечня ниже в соответствии с документацией.

В случае использования прочих вызовов необходимо производить разработку отдельного СКЗИ в соответствии с действующей нормативной базой (в частности, с Постановлением Правительства Российской Федерации от 16 апреля 2012 г. №313 и Положением о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации (Положение ПКЗ-2005)).

Метод	Описание	Ограничения на использование метода
`getInstance()` класса `KeyPairGenerator`	Метод создает объект генерации ключевой пары ЭП и VKO (генератор).	
`getInstance()` класса `KeyFactory`	Метод создаёт объект, создающий объекты закрытых и открытых ключей ЭП и VKO на основе ключевых спецификаций, представляющих собой внутреннее представление ключевой информации.	
`getInstance()` класса `KeyStore`	Метод создает объект, осуществляющий доступ к ключевому хранилищу.	
`getInstance()` класса `SecureRandom`	Метод создает объект класса генератора случайных чисел.	
`getInstance()` класса `Signature`	Метод создает объект формирования ЭП в соответствии с указанным идентификатором алгоритма подписи.	
`getInstance()` класса `MessageDigest`	Метод создает объект функции хэширования в соответствии с указанным идентификатором алгоритма хэширования.	
`getInstance()` класса `KeyAgreement`	Метод создает объект, вырабатывающий ключи согласования.	

`getInstance()` класса `Mac`	Метод создает объект, осуществляющий имитозащиту данных по алгоритму, соответствующему указанному идентификатору.	Разрешен при указании параметра HMAC_GOSTR3411, HMAC_GOSTR3411_2012_256 или HMAC_GOSTR3411_2012_512.
`Initialize()` класса `KeyPairGenerator`	Метод осуществляет операцию изменения набора параметров вырабатываемых ключевых пар ЭП или VKO (алгоритм ЭП/VKO, параметры используемой группы точек эллиптической кривой, алгоритм хэширования, узла замены для алгоритма хэширования).	
`generateKeyPair()` класса `KeyPairGenerator`	Метод осуществляет генерацию ключевой пары.	
`generatePublic()` класса `KeyFactory`	Метод создает объект открытого ключа на основе спецификации открытого ключа (объекта класса PublicKeySpec), содержащей ключевой материал, или закодированного ключа (в виде объекта класса X509EncodedKeySpec).	
`generatePrivate()` класса `KeyFactory`	Метод создаёт объект закрытого ключа/ключа ЭП на основе спецификации закрытого ключа (объекта класса PrivateKeySpec), содержащей ключевой материал.	
`Load()` класса `KeyStore`	Метод осуществляет загрузку содержимого стандартного хранилища JCA в память.	
`getName()` класса `KeyStore`	Метод возвращает имя хранилища JCA.	
`getEntry()` класса `KeyStore`	Метод возвращает дескриптор ключа для дальнейшей работы с ключом при указании верного пароля.	
`setKeyEntry()` класса `KeyStore`	Метод осуществляет запись закрытого ключа/ключа ЭП на носитель.	
`store()` класса `KeyStore`	Метод осуществляет перезапись стандартного хранилища JCA.	
`getKey()` класса `KeyStore`	Метод осуществляет чтение ключа ЭП с носителя.	

`setCertificateEntry()` класса `KeyStore`	Метод осуществляет запись сертификата ключа проверки ЭП на носитель, запись сертификата в хранилище.	
`getCertificate()` класса `KeyStore`	Метод осуществляет чтение сертификата ключа проверки ЭП с носителя, чтение сертификата из хранилища.	
`deleteEntry()` класса `KeyStore`	Метод производит удаление ключевого контейнера с носителя.	
`reset()` класса `MessageDigest`	Метод переинициализирует объект хэширования для повторного использования после получения хэш-значения предыдущего сообщения. Метод может также изменять параметры хэширования (OID) (при использовании алгоритма хэширования ГОСТ Р 34.11-94).	
`Clone()` класса `MessageDigest`	Метод создает точную копию существующего объекта хэширования.	
`Update()` класса `MessageDigest`	Метод осуществляет обработку хэшируемых данных.	
`read()` класса `DigestInputStream`	Метод осуществляет обработку хэшируемых данных, получаемых из потока.	
`digest()` класса `MessageDigest`	Метод завершает операцию хэширования и получает хэш-значение сообщения.	
`valid()` класса `PrivateKeyUsageExtension`	Метод, проверяющий срок действия ключа электронной подписи.	
`initSign()` класса `Signature`	Метод устанавливает ключ ЭП и параметры ЭП.	
`initVerify()` класса `Signature`	Метод устанавливает ключ проверки электронной подписи и параметры ЭП.	Метод разрешается использовать только для объектов открытых ключей, полученных с помощью механизмов PKIX.
`setParameter()` класса `Signature`	Метод устанавливает используемую хэш-функцию и её параметры в соответствии с переданным идентификатором.	
`update()` класса `Signature`	Метод осуществляет обработку переданных данных хэш-функцией.	

`sign()` класса `Signature`	Метод производит завершающее преобразование хэш-функции и производит подпись всего накопленного сообщения.	
`verify()` класса `Signature`	Метод производит завершающее преобразование хэш-функции и осуществляет проверку подписи.	
`generateCertificate()` класса `CertificateFactory`	Метод производит генерацию X.509 сертификатов.	
`getEncoded()` класса `Certificate`, наследуется `X509Certificate`	Метод осуществляет кодирование существующего сертификата.	
`getPublicKey()` класса `Certificate`, наследуется `X509Certificate`	Метод возвращает ключ проверки ЭП из сертификата.	
`checkValidity()` класса `X509Certificate`	Метод осуществляет проверку срока действия сертификата X509.	
`getVersion()` класса `X509Certificate`	Метод возвращает номер версии сертификата X509.	
`getSerialNumber()` класса `X509Certificate`	Метод возвращает серийный номер сертификата X509.	
`getIssuerDN()` класса `X509Certificate`	Метод возвращает DN эмитента сертификата.	
`getIssuerX500Principal()` класса `X509Certificate`	Метод возвращает DN эмитента сертификата в формате X500.	
`getSubjectDN()` класса `X509Certificate`	Метод возвращает DN владельца сертификата.	
`getSubjectX500Principal()` класса `X509Certificate`	Метод возвращает DN владельца сертификата в формате X500.	
`getNotBefore()` класса `X509Certificate`	Метод возвращает дату начала действия сертификата.	
`getNotAfter()` класса `X509Certificate`	Метод возвращает дату окончания действия сертификата.	
`getTBSCertificate()` класса `X509Certificate`	Метод осуществляет DER-кодирование сертификата.	
`getSignature()` класса `X509Certificate`	Метод возвращает подпись под сертификатом.	
`getSigAlgName()` класса `X509Certificate`	Метод возвращает название алгоритма подписи под сертификатом.	
`getSigAlgOID()` класса `X509Certificate`	Метод возвращает OID алгоритма подписи под сертификатом.	

`getSigAlgParams()` класса `X509Certificate`	Метод возвращает параметры алгоритма подписи под сертификатом в DER-кодировке.	
`getIssuerUniqueID()` класса `X509Certificate`	Метод возвращает уникальный ID эмитента сертификата.	
`getSubjectUniqueID()` класса `X509Certificate`	Метод возвращает уникальный ID владельца сертификата.	
`getKeyUsage()` класса `X509Certificate`	Метод возвращает набор разрешенных областей использования ключа.	
`getExtendedKeyUsage()` класса `X509Certificate`	Метод возвращает области расширенного использования ключа.	
`getBasicConstraints()` класса `X509Certificate`	Метод возвращает длину расширения BasicConstraints.	
`getIssuerAlternativeNames()` класса `X509Certificate`	Метод возвращает список альтернативных имён эмитента сертификата.	
`getSubjectAlternativeNames()` класса `X509Certificate`	Метод возвращает список альтернативных имён владельца сертификата.	
`getOID()` интерфейса `ParamsInterface`	Метод возвращает объектный идентификатор параметров алгоритма ЭП/VKO/хэширования/узлов замены шифрования.	
`getDefault()` интерфейса `ParamsInterface`	Метод возвращает значение объектного идентификатора, установленного в контрольной панели, параметров алгоритма ЭП/VKO/хэширования/узлов замены шифрования.	
`setDefaultAvailable()` интерфейса `ParamsInterface`	Метод осуществляет проверку наличия необходимых прав для установки новых параметров по умолчанию в контрольную панель.	
`setDefault(OID def)` интерфейса `ParamsInterface`	Метод устанавливает объектный идентификатор по умолчанию для параметров алгоритма ЭП/VKO/хэширования/узлов замены шифрования.	

`getOIDs()` интерфейса `ParamsInterface`	Метод получает список допустимых объектных идентификаторов параметров для алгоритма ЭП/VKO/хэширования/узлов замены шифрования.	
`getDefaultSignParams()` класса `AlgIdSpec`	Метод возвращает параметры алгоритма подписи по умолчанию.	
`getDefaultDigestParams()` класса `AlgIdSpec`	Метод возвращает параметры алгоритма хэширования по умолчанию.	
`getDefaultCryptParams()` класса `AlgIdSpec`	Метод возвращает параметры алгоритма шифрования по умолчанию.	
`setKeyUsage()` класса `GostCertificateRequest`	Метод устанавливает значение поля keyUsage в запросе на сертификат, описывающее разрешенные области использования ключа.	
`addExtKeyUsage()` класса `GostCertificateRequest`	Метод устанавливает значение поля extKeyUsage в запросе на сертификат, описывающее дополнительные области использования ключа.	
`addExtension()` класса `GostCertificateRequest`	Метод устанавливает дополнительное расширение в список расширений.	
`setPublicKeyInfo()` класса `GostCertificateRequest`	Метод осуществляет кодирование и запись в структуру запроса параметров и значения передаваемого открытого ключа субъекта.	
`setSubjectInfo()` класса `GostCertificateRequest`	Метод определяет имя субъекта в формате X.500, устанавливает новое имя субъекта.	
`encodeAndSign()` класса `GostCertificateRequest`	Метод выполняет кодирование запроса в DER-кодировку и подпись сертификата.	
`getEncodedCert()` класса `GostCertificateRequest`	Метод отправляет запрос центру сертификации и получает соответствующий запросу сертификат.	

`getEncodedCertFromDER()` класса `GostCertificateRequest`	Метод отправляет запрос в DER-кодировке центру сертификации и получает соответствующий запросу сертификата.	
`printToDER()` класса `GostCertificateRequest`	Метод осуществляет печать подписанного запроса в DER-кодировке в передаваемый PrintStream.	
`getEncodedCertFromBASE64()` класса `GostCertificateRequest`	Метод осуществляет отправку запроса в кодировке BASE64 центру сертификации и получение соответствующего запросу сертификата.	
`printToBASE64()` класса `GostCertificateRequest`	Метод осуществляет печать подписанного запроса в BASE64-кодировке в передаваемый PrintStream.	
`getEncodedRootCert()` класса `GostCertificateRequest`	Метод запрашивает и получает корневой сертификат центра сертификации.	
`getRootCertList()` класса `CA15GostCertificateRequest`	Метод получает список корневых сертификатов УЦ (CA15).	
`sendCertificateRequest ()` класса `CA15GostCertificateRequest`	Метод отправляет в УЦ запрос на сертификат.	
`getCertificateRequestList ()` класса `CA15GostCertificateRequest`	Метод получает список запросов на сертификат и статус их обработки.	
`checkCertificateStatus ()` класса `CA15GostCertificateRequest`	Метод получает статус обработки УЦ ранее отправленного запроса на сертификат.	
`getCertificateRequestId()` класса `CA15GostCertificateRequest`	Метод получает строки-идентификаторы запроса на сертификат.	
`getCertificateByRequestId()` класса `CA15GostCertificateRequest`	Метод получает выпущенный сертификат по идентификатору запроса.	
`init()` класса `Mac`	Метод, устанавливающий алгоритм шифрования.	
`clone()` класса `Mac`	Метод, производящий копирование объекта имитозащиты.	
`update()` класса `Mac`	Метод, осуществляющий обработку данных в процессе вычисления имитовставки.	

doFinal() класса Mac	Метод, завершающий вычисление значения имитовставки.	
reset() класса Mac	Метод, осуществляющий переинициализацию объекта класса Mac после окончания вычисления значения имитовставки предыдущего сообщения.	
verify() класса CAdESSignature	Метод осуществляет проверку всех подписей CAdES.	
getCAdESSignerInfos() класса CAdESSignature	Метод получает список всех подписантов.	
setCertificateStore() класса CAdESSignature	Метод устанавливает набор сертификатов, которые будут упакованы в подписанное сообщение.	
setCRLStore() класса CAdESSignature	Метод устанавливает набор списков отозванных сертификатов, которые будут упакованы в подписанное сообщение.	
addSigner() класса CAdESSignature	Метод осуществляет добавление подписи в формируемое подписанное сообщение.	
open() класса CAdESSignature	Метод осуществляет открытие потока подписываемых данных.	
update() класса CAdESSignature	Метод осуществляет обработку порции данных.	
close() класса CAdESSignature	Метод осуществляет завершение обработки подписываемых данных.	
replaceSigners() класса CAdESSignature	Метод, осуществляющий изменения данных подписывающих в упакованном подписанном сообщении.	
addRecipient() класса EnvelopedSignature	Метод, добавляющий информацию о получателе CMS сообщения, и подготавливающий сообщение к зашифрованию.	
addKeyTransRecipient() класса EnvelopedSignature	Метод добавляет переданный сертификат в список информации о получателе CMS сообщения в виде структуры key_transport.	Метод разрешается использовать только при осуществлении проверки цепочки сертификата с помощью механизмов PKIX и проверки области действия сертификата.

addKeyAgreeRecipient() класса EnvelopedSignature	Метод добавляет переданный сертификат в список информации о получателе CMS сообщения в виде структуры key_agreement.	Метод разрешается использовать только при осуществлении проверки цепочки сертификата с помощью механизмов PKIX и проверки области действия сертификата.
open() класса EnvelopedSignature	Метод, осуществляющий открытие потока зашифрования CMS сообщения.	
update() класса EnvelopedSignature	Метод, осуществляющий шифрование порции данных CMS сообщения.	
close() класса EnvelopedSignature	Метод, осуществляющий шифрование финальной порции данных и закрывающий поток зашифрования CMS сообщения.	
getRecipients() класса EnvelopedSignature	Метод, возвращающий список получателей CMS сообщения.	
decrypt() класса EnvelopedSignature	Метод, позволяющий получателю расшифровать CMS сообщение.	
getCAdESSignatureType() класса CAdESType	Метод возвращает тип CAdES сообщения.	
getSignerInfo() класса CAdESSigner	Метод возвращает информацию о подписанте.	
getSignerSignedAttributes() класса CAdESSigner	Метод возвращает список подписываемых атрибутов сообщения.	
getSignerUnsignedAttributes() класса CAdESSigner	Метод возвращает список неподписываемых атрибутов сообщения.	
getSignatureTimestampToken() класса CAdESSigner	Метод возвращает внутренний штамп (signature-timestamp) времени сообщения.	
getCAdESCTimestampToken() класса CAdESSigner	Метод возвращает внешний штамп (CAdES-C-timestamp) времени сообщения.	
getCAdESCertificates() класса CAdESSigner	Метод возвращает список сертификатов (certificate-values).	
getSignerCertificate() класса CAdESSigner	Метод возвращает сертификат подписавшего сообщение.	
getSignatureType() класса CAdESSigner	Метод возвращает тип подписи.	
getCAdESCountersignerInfos() класса CAdESSigner	Метод возвращает список заверителей сообщения.	

`setCertificateStore()` класса `CAdESSigner`	Метод устанавливает хранилище сертификатов, из которого позже может быть получен сертификат подписи.	
`enhance()` класса `CAdESSigner`	Метод осуществляет «усовершенствование» подписи CAdES-BES до CAdES-X Long Type 1.	
`addCountersigner()` класса `CAdESSigner`	Метод, осуществляющий добавление заверяющей подписи к отдельному подписанту.	
`verify()` класса `CAdESSigner`	Метод, осуществляющий проверку одной отдельной подписи CAdES.	
`build()` класса `CertPathBuilder`	Метод, осуществляющий построение цепочки сертификатов.	
`getAlgorithm()` класса `CertPathValidator`	Метод, возвращающий имя алгоритма, осуществляющего проверку цепочек сертификатов.	
`getDefaultType()` класса `CertPathValidator`	Метод, возвращающий имя алгоритма, осуществляющего проверку цепочек сертификатов, принятого по умолчанию.	
`getInstance()` класса `CertPathValidator`	Метод, возвращающий объект класса CertPathValidator.	
`getProvider()` класса `CertPathValidator`	Метод, осуществляющий получение объекта провайдера алгоритма, осуществляющего проверку цепочек сертификатов, принятого по умолчанию.	
`validate()` класса `CertPathValidator`	Метод, осуществляющий проверку цепочек сертификатов.	
`addCertPathChecker()` класса `CertPathParameters`	Метод, устанавливающий дополнительные правила проверки сертификатов.	
`addCertStore()` класса `CertPathParameters`	Метод, устанавливающий дополнительные хранилища сертификатов и CRL.	
`getCertPathCheckers()` класса `CertPathParameters`	Метод, возвращающий список установленных дополнительных правил проверки сертификатов.	
`getCertStores()` класса `CertPathParameters`	Метод, возвращающий список хранилищ сертификатов и CRL.	

`getDate()` класса `CertPathParameters`	Метод, возвращающий дату, на которую проверяется верность цепочки сертификатов.	
`getInitialPolicies()` класса `CertPathParameters`	Метод, возвращающий список OID'ов политик, которые должны быть указаны в сертификатах, из которых строится цепочка.	
`getPolicyQualifiersRejected()` класса `CertPathParameters`	Метод, возвращающий флаг PolicyQualifiersRejected.	
`getSigProvider()` класса `CertPathParameters`	Метод, получающий имя провайдера ЭП.	
`getTargetCertConstraints()` класса `CertPathParameters`	Метод, возвращающий ограничения на целевой сертификат.	
`getTrustAnchors()` класса `CertPathParameters`	Метод, получающий список доверенных сертификатов.	
`isAnyPolicyInhibited()` класса `CertPathParameters`	Метод, возвращающий признак обработки всех политик, указанных в сертификате.	
`isExplicitPolicyRequired()` класса `CertPathParameters`	Метод, возвращающий признак наличия явных описаний политик.	
`isPolicyMappingInhibited()` класса `CertPathParameters`	Метод, возвращающий признак подавления PolicyMapping.	
`isRevocationEnabled()` класса `CertPathParameters`	Метод, устанавливающий признак необходимости проверки сертификатов с помощью OCSP/CRL.	
`setAnyPolicyInhibited()` класса `CertPathParameters`	Метод, устанавливающий признак обработки всех политик, указанных в сертификате.	
`setCertPathCheckers()` класса `CertPathParameters`	Метод, устанавливающий набор дополнительных правил проверки сертификатов.	
`setCertStores()` класса `CertPathParameters`	Метод, устанавливающий набор дополнительных хранилищ сертификатов и CRL.	
`setDate()` класса `CertPathParameters`	Метод, устанавливающий дату, на которую должна производиться проверка цепочки сертификатов.	
`setExplicitPolicyRequired()` класса `CertPathParameters`	Метод, устанавливающий признак наличия явных описаний политик.	

`setInitialPolicies()` класса `CertPathParameters`	Метод, устанавливающий список OID'ов политик, которые должны быть указаны в сертификатах, из которых строится цепочка.	
`setPolicyMappingInhibited()` класса `CertPathParameters`	Метод, устанавливающий признак подавления PolicyMapping.	
`setPolicyQualifiersRejected()` класса `CertPathParameters`	Метод устанавливает флаг PolicyQualifiersRejected.	
`setRevocationEnabled()` класса `CertPathParameters`	Метод, устанавливающий признак необходимости проверки сертификатов с помощью OCSP/CRL.	
`setTargetCertConstraints()` класса `CertPathParameters`	Метод, устанавливающий ограничения на целевой сертификат.	
`setTrustAnchors()` класса `CertPathParameters`	Метод, устанавливающий список доверенных сертификатов.	
`getMaxPathLength()` класса `PKIXBuilderParameters`	Метод, получающий максимальную возможную длину строимой цепочки.	
`setMaxPathLength()` класса `PKIXBuilderParameters`	Метод, устанавливающий максимальную возможную длину строимой цепочки.	
`getXAdESSignerInfos()` класса `XAdESSIgnature`	Метод получает список всех подписантов.	
`addSigner()` класса `XAdESSIgnature`	Метод осуществляет добавление подписи в формируемое подписанное XML сообщение.	
`open()` класса `XAdESSIgnature`	Метод осуществляет открытие потока подписываемых данных.	
`update()` класса `XAdESSIgnature`	Метод осуществляет обработку порции данных.	
`close()` класса `XAdESSIgnature`	Метод осуществляет завершение обработки подписываемых данных.	
`verify()` класса `XAdESSIgnature`	Метод осуществляет проверку всех подписей XAdES.	
`getSignatureType()` класса `XAdESSigner`	Метод, возвращающий тип XAdES-подписи.	
`getEarliestValidSignatureTime-StampToken()` класса `XAdESSignerT`	Метод, возвращающий самый ранний валидный внутренний штамп времени.	

`verify()` класса `XAdESSigner`	Метод, осуществляющий проверку одной отдельной подписи XAdES.	
`getSignerInfo()` класса `XAdESSigner`	Метод, возвращающий узел подписи в документе.	
`getSignerCertificate()` класса `XAdESSigner`	Метод, возвращающий сертификат ключа проверки ЭП данного подписанта.	
`getElement()` класса `XAdESSigner`	Метод, возвращающий подписываемый узел.	
`getDocument()` класса `XAdESSigner`	Метод, возвращающий подписываемый документ.	
`getPrivateKey()` класса `JCPPrivateKeyEntry`	Метод, возвращающий дескриптор ключа privKey.	
`isExportable()` класса `JCPPrivateKeyEntry`	Метод, возвращающий значение флага экспортируемости закрытого ключа/ключа ЭП.	
`getCertificateChain()` класса `JCPPrivateKeyEntry`	Метод, возвращающий цепочку сертификатов.	
`getCertificate()` класса `JCPPrivateKeyEntry`	Метод, возвращающий клиентский сертификат (первый в цепочке сертификатов).	
`toString()` класса `JCPPrivateKeyEntry`	Метод, возвращающий строковое представление цепочки сертификатов.	
`isSilentMode()` класса `JCPProtectionParameter`	Метод, возвращающий режим открытия ключевого контейнера.	
`isAllowEmptyChain()` класса `JCPProtectionParameter`	Метод, возвращающий разрешение использовать null вместо цепочки сертификатов.	
`getKeyType()` класса `JCPProtectionParameter`	Метод, возвращающий тип ключа.	
`sign()` класса `XMLSignature`	Метод создает ЭП сообщения.	Разрешено использование только со следующими URI/URN: • http://www.w3.org/2001/04/ xmldsig-more#gostr34102001-gostr3411 • urn:ietf:params:xml:ns:cpxmlsec: algorithms:gostr34102001-gostr3411 • urn:ietf:params:xml:ns:cpxmlsec: algorithms:gostr34102012-gostr34112012-256 • urn:ietf:params:xml:ns:cpxmlsec: algorithms:gostr34102012-gostr34112012-512

`checkSignatureValue() класса XMLSignature`	Метод осуществляет проверку ЭП сообщения.	Метод разрешается использовать только при осуществлении проверки сертификата открытого ключа с помощью механизмов PKIX.
Методы установки одностороннего TLS-соединения		
`initClientSSL объекта ru.CryptoPro.ssl.util.TLSContext`	Метод создания и инициализации объекта SSLContext для клиентской стороны TLS-соединения с односторонней аутентификацией.	Разрешён при указании в качестве параметра tlsProvider значения «JTLS». Перед использованием системная переменная «tls_prohibit_disabled_validation» должна быть установлена в значение «True».
`getSocketFactory объекта SSLContext`	Метод создания объекта SSLSocketFactory на основе текущегоSSLContext.	Разрешён только для объекта SSLContext, полученного разрешённым вызовом initClientSSL объекта ru.CryptoPro.ssl.util.TLSContext.
`createSocket объекта SSLSocketFactory`	Метод создания программного сокета с указанием физического адреса сервера.	Разрешён только для объекта SSLSocketFactory, полученного комбинацией разрешённых вызовов TLSContext.initClientSSL и SSLContext.getSocketFactory. Если createSocket был вызван без параметров, то для полученного объекта SSLSocket нужно вызвать его метод connect.
`connect объектаSSLSocket`	Метод для установки связи сокета с сервером по указанному адресу.	Разрешён только для объекта SSLSocket, полученного комбинацией разрешённых вызовов TLSContext.initClientSSL, SSLContext.getSocketFactory и SSLSocketFactory.createSocket.
`startHandshake объекта SSLSocket`	Метод для установки TLS-соединения или проведения RENEGOTIATION.	Разрешён только для объекта SSLSocket, полученного комбинацией разрешённых вызовов TLSContext.initClientSSL, SSLContext.getSocketFactory и SSLSocketFactory.createSocket.
`getOutputStream объекта SSLSocket`	Метод для получения доступа к буферу OutputStream для отправки данных TLS-серверу.	Разрешён только для объекта SSLSocket, полученного комбинацией разрешённых вызовов TLSContext.initClientSSL, SSLContext.getSocketFactory и SSLSocketFactory.createSocket.

`getInputStream объекта SSLSocket`	Метод для получения доступа к буферу InputStream для получения данных от TLS-сервера.	Разрешён только для объекта SSLSocket, полученного комбинацией разрешённых вызововTLSContext.initClientSSL, SSLContext.getSocketFactory и SSLSocketFactory.createSocket.
`write объекта OutputStream`	Метод для отправки данных TLS-серверу.	Разрешён только для объекта OutputStream, полученного комбинацией разрешённых вызовов TLSContext.initClientSSL, SSLContext.getSocketFactory, SSLSocketFactory.createSocket и SSLSocket.getOutputStream. Значение провайдера по умолчанию должно быть равно «JCP»; для установки допустимо использовать опцию в панели управления JCP и вызовы cpSSLConfig. setDefaultSSLProvider("JCP") и System.setProperty ("ru.CryptoPro.defaultSSLProv "JCP").
`read объекта InputStream`	Метод для получения данных от TLS-сервера.	Разрешён только для объекта InputStream, полученного комбинацией разрешённых вызововTLSContext.initClientSSL, SSLSocketFactory.createSocket и SSLSocket.getInputStream. Значение провайдера по умолчанию должно быть равно «JCP»; для установки допустимо использовать опцию в панели управления JCP и вызовы cpSSLConfig. setDefaultSSLProvider("JCP") и System.setProperty ("ru.CryptoPro.defaultSSLProv "JCP").
`close объектаSSLSocket`	Метод для закрытия TLS-соединения.	Разрешён только для объекта SSLSocket, полученного комбинацией разрешённых вызовов TLSContext.initClientSSL, SSLContext.getSocketFactory и SSLSocketFactory.createSocket.

Приложение 3
Приложение для создания CMS сообщений

Утилита `cmsutil` предназначена для выработки CMS-enveloped сообщений в соответствии с RFC 4490 и рекомендациями МР 26.2.002-2013 «Информационная технология. Криптографическая защита информации. Использование алгоритмов ГОСТ 28147-89, ГОСТ Р 34.10 и ГОСТ Р 34.11 в криптографических сообщениях формата CMS».

Для **зашифрования файла** с помощью утилиты `cmsutil` необходимо выполнить следующую команду:

```
java -jar cmsutil.jar -encrypt
-certstore <Название хранилища доверенных сертификатов>
[-pass <Пароль от хранилища>] -alias <Имя сертификата>
-in <Файл с данными для шифрования> -out <Файл выхода>
```

Для **расшифрования файла** с помощью утилиты `cmsutil` необходимо выполнить следующую команду:

```
java -jar cmsutil.jar -decrypt
  -keystore <Тип хранилища>
-alias <Имя контейнера> [-pass <Пароль контейнера>]
-in <Файл с шифрованными данными> -out <Файл выхода>
```

Также возможно использовать следующий вызов с аналогичными наборами команд (при этом необходимые библиотеки будут подключены из текущей папки):

```
java -cp * cmsutil.CMSMain [параметры]
```

Приложение 4
Приложение для создания TLS-туннеля

Утилита `tls_proxy` представляет собой приложение для создания TLS-туннеля. Данная утилита позволяет устанавливать TLS-соединение в соответствии с рекомендациями МР 26.2.001-2013 «Информационная технология. Криптографическая защита информации. Использование наборов алгоритмов шифрования на основе ГОСТ 28147-89 для протокола безопасности транспортного уровня (TLS)».

Утилита может быть запущена следующим образом:

```
tls_proxy <ListenPort>
```

При этом в папке приложения должен лежать конфигурационный файл, устроенный следующим образом:

```
<Config>
  <Parameters inactiveTimeout=«60» checkInactiveTimeout=«30» serverSoTimeout=«600» provider="JCP"
  protocol="GostTLS" />
  <CertStore path=«c:\software\Keys\tomcat7\test_ca.store» password=«1» type="CertStore"
  provider="JCP" />
    <Addresses>
      <Address>
        <ListenPort>9000</ListenPort>
        <Host>cpca.cryptopro.ru</Host>
        <Port>443</Port>
        <ClientAuthEnabled>false</ClientAuthEnabled>
      </Address>
      <Address>
        <ListenPort>9001</ListenPort>
        <Host>cryptopro.ru</Host>
        <Port>4444</Port>
        <ClientAuthEnabled>true</ClientAuthEnabled>
        <KeyType>HDImageStore</KeyType>
        <KeyPassword>Pass1234</KeyPassword>
      </Address>
    </Addresses>
</Config>
```

Parameters	технологические параметры соединений:
inactiveTimeout	период времени (сек), после которого подключение считается неактивным и может быть закрыто (время активности обновляется при каждом запросе)
checkInactiveTimeout	период (сек) проверки соединений (для таймера); неактивные соединения закрываются
serverSoTimeout	период ожидания (сек) входящих соединений
provider	провайдер, реализующий работу с хранилищем, алгоритмом; должен принимать значение только «JCP» или «JCSP»
protocol	протокол соединения; должен принимать значение только «GostTLS»
CertStore	хранилище доверенных сертификатов:
path	путь к хранилищу

password	пароль к хранилищу
type	тип хранилища
provider	провайдер, реализующий хранилище
Addresses	блок адресов
Address	отдельный адрес сервера
ListenPort	порт, данные из которого транслируются на сервер и обратно. С этим портом запускается приложение tls_proxy. Отдельный экземпляр приложения, слушающего ListenPort, работает с конкретным адресом, связанным с этим портом.
Host, Port	параметры сервера, куда происходит трансляция из ListenPort
ClientAuthEnabled	флаг, определяющий, включена ли на сервере клиентская аутентификация; если включена, то потребуются параметры KeyType и KeyPassword
KeyType	тип контейнера
KeyPassword	пароль контейнера