Утверждён

КБДЖ.468243.218 ПП-ЛУ

СРЕДСТВО КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ (СКЗИ)

«РУТОКЕН TLS»

Исполнение 1

Правила пользования

КБДЖ.468243.218 ПП

Изделие «Средство криптографической защиты информации «РУТОКЕН TLS». Исполнение 1»

Правила пользования КБДЖ.468243.218 ПП

© АО «Актив-софт»

Редакция от августа 2025 г. (введена извещением об изменении КБДЖ.907-25)

АННОТАЦИЯ

В данном документе изложены правила пользования средством криптографической защиты информации (СКЗИ) «РУТОКЕН TLS». Исполнение 1». При использовании СКЗИ необходимо также следовать требованиям следующих нормативных и эксплуатационных документов:

- СКЗИ «Рутокен TLS» Исполнение 9 (далее – «АРМ ЗКИ «РУТОКЕН TLS»). Правила пользования (КБДЖ.468243.217 ПП);
- Правила пользования и формуляр на МДЗ из состава СКЗИ «Рутокен TLS» Исполнение 4 (при использовании по классу КС2).

В состав СКЗИ «РУТОКЕН TLS» входит:

- Аппаратный модуль (АМ) «РУТОКЕН TLS». Исполнение 1 (КБДЖ.468243.218);
- ПО для ОС, обеспечивающее прикладной программный интерфейс доступа к АМ по стандарту PKCS#11 (КБДЖ.01456);
- Модуль сетевой аутентификации (КБДЖ.01454);
- Модуль контроля целостности (КБДЖ.01455);
- Модуль доверенной загрузки имеющий действующий сертификат по требованиям ФСБ России (для уровня защиты КС2).

ОГЛАВЛЕНИЕ

1. Общие положения 5
2. Назначение и основные характеристики СКЗИ «РУТОКЕН TLS» 5
2.1. Программно-аппаратные среды функционирования 6
3. Требования по обеспечению безопасности применения СКЗИ 7
3.1. Требования безопасности функционирования рабочих мест 7
3.2. Специальные требования 9
4. Требования по эксплуатации СКЗИ 10
4.1. Общие правила эксплуатации 10
4.2. Требования по обеспечению целостности СКЗИ 10
4.2.1. Средства контроля целостности 11
4.3. Требования к аутентификации и разграничению доступа 12
4.4. Ввод СКЗИ в эксплуатацию 12
4.5. Действия по восстановлению работоспособности в нештатных ситуациях 13
4.6. Порядок выполнения технического обслуживания, ремонт и утилизация 13
4.7. Требования к встраиванию 14
4.8. Подготовка АМ к эмиссии 14
5. Ключевые носители СКЗИ «Рутокен TLS» Исполнение 1 14
5.1. Ключевые носители 14
5.2. Хранение ключевых носителей 14
5.3. Сроки действия ключевой информации 15
5.4. Плановая смена ключевой информации 15
5.5. Компрометация ключевых носителей 15
5.6. Уничтожение ключевой информации и ключевых носителей 16
5.7. Ведение журналов операций 16
Приложение 1 18
Приложение 2 28

1. Общие положения

СКЗИ «Рутокен TLS» Исполнение 1 реализует функции выполнения и проверки электронной подписи только в автоматическом режиме.

Сертификация СКЗИ «Рутокен TLS» Исполнение 1 распространяется на использование отечественных криптографических алгоритмов: ГОСТ 28147–89, ГОСТ Р 34.10–2012, ГОСТ 34.10–2018, ГОСТ Р 34.11–2012, ГОСТ 34.11–2018.

СКЗИ представляет собой набор программного обеспечения (ПО), которое функционирует на ПЭВМ пользователя, и специальное программное обеспечение (СПО), функционирующее на аппаратном модуле (АМ), для обеспечения защиты конфиденциальной информации, в том числе персональных данных, на отдельных рабочих местах.

СКЗИ «Рутокен TLS» Исполнение 1 предназначено для эксплуатации на территории Российской Федерации.

Период непрерывной работы «Рутокен TLS» Исполнение 1 не должен превышать 1 суток.

Транспортировка, учет, хранение, размещение, техническое обслуживание, эксплуатация, уничтожение экземпляров СКЗИ «Рутокен TLS» Исполнение 1, технической и ключевой документации к ним, а также допуск к аппаратуре и ключевым документам должны осуществляться в строгом соответствии со следующими документами:

- «Инструкция об организации и обеспечении безопасности хранения, обработки и передачи по каналам связи с использованием средств криптографической защиты информации с ограниченным доступом, не содержащей сведений, составляющих государственную тайну» (Приказ ФАПСИ № 152 от 13 июня 2001 года)
- «Положение о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации» (Приказ ФСБ России № 66 от 9 февраля 2005 года)
- эксплуатационной документацией, указанной в п. 2.4 ФО СКЗИ «Рутокен TLS» Исполнение 1.

2. Назначение и основные характеристики СКЗИ «РУТОКЕН TLS»

СКЗИ «РУТОКЕН TLS» подключается по шине USB к ПЭВМ и предназначено для шифрования, аутентификации, автоматического вычисления и проверки электронной подписи (ЭП), установки TLS-соединения и безопасного хранения данных.

СКЗИ «РУТОКЕН TLS» обеспечивает:

- хранение во внутренней памяти информации, передаваемой в изделие через порт USB ПЭВМ;
- импорт ключей проверки ЭП, ключей ЭП, доверенных сертификатов и ключей шифрования при помощи АРМ ЗКИ «РУТОКЕН TLS»;
- аутентификацию для доступа к объектам файловой системы АМ «Рутокен TLS»;
- выработку псевдослучайных последовательностей при помощи ПДСЧ АМ «Рутокен TLS»;
- выработку ключей шифрования внутри изделия и их защищенное хранение;
- выработку ключей ЭП, проверки ЭП внутри изделия и их защищенное хранение;
- выработку сессионных ключей (ключей парной связи) в соответствии с Р 50.1.113-2016 (VKO_GOSTR3410_2012_256, VKO_GOSTR3410_2012_512);

- выработку производных ключей в соответствии с Р 50.1.113-2016 (KDF_TREE_GOSTR3411_2012_256);
- экспорт и импорт сессионных ключей;
- вычисление и проверку ЭП в автоматическом режиме в соответствии с ГОСТ Р 34.10–2012, ГОСТ 34.10–2018;
- вычисление значения хэш-функции в соответствии с ГОСТ Р 34.11–2012, ГОСТ 34.11–2018;
- вычисление и проверку кода аутентификации сообщения HMAC на основе хэш-функции в соответствии с Р 50.1.113-2016 (HMAC_GOSTR3411_2012_256, HMAC_GOSTR3411_2012_512);
- шифрование и расшифрование данных в соответствии с ГОСТ 28147–89 в режимах простой замены и гаммирования с обратной связью;
- вычисление и проверку имитовставки в соответствии с ГОСТ 28147–89;

Примечание – при встраивании СКЗИ «Рутокен TLS» Исполнение 1 и использовании функций шифрования или имитозащиты данных необходимо обеспечить защиту от повторов сообщений сторонними средствами (например, введение в сообщение метки времени создания и проверки ее получателем сообщения).

- идентификацию, аутентификацию, шифрование и имитозащиту TLS-соединений в соответствии с Р 1323565.1.020-2020.

2.1. Программно-аппаратные среды функционирования

Интерфейсная библиотека PKCS#11, модуль контроля целостности, модуль сетевой аутентификации функционируют на следующих программно-аппаратных платформах:

– Поддерживаемые ОС:

- Windows 7/8/Server 2012/8.1/Server 2012 R2/10/11/Server 2016/Server 2019 /Server 2022.
- Linux (LSB 4.0, 4.1, 5.0; Ubuntu 20.04 / 22.04 / 24.04; Debian 10 / 11 / 12; RHEL 6 / 7 / 8 / 9; Fedora 37 / 38 / 39; CentOS 7 /8; Альт СП релиз 10; Альт 8 СП; Альт Рабочая станция 9 / 10 / К 9 / К 10; Альт Сервер 9 / 10; Альт Образование 9 / 10; Oracle Enterprise Linux 6 / 7 / 8 / 9; OpenSUSE Leap 15; SUSE Linux Enterprise 12 / 15; Astra Linux Special Edition РУСБ.10015-16 (исп. 1 и 2) / РУСБ.10015-17 / РУСБ.10015-01 / РУСБ.10152-02 / РУСБ.10265-01; Astra Linux Common Edition ТУ 5011-001-88328866-2008; FreeBSD 12 / 13 / 14; РЕД ОС 7.3 / 8; РОСА «ХРОМ» рабочая станция; РОСА «ХРОМ» сервер; РОСА «КОБАЛЬТ» рабочая станция; РОСА «КОБАЛЬТ» сервер; РОСА «БАРИЙ»; РОСА «КОБАЛЬТ» рабочая станция, сертифицированная ФСТЭК России; Гослинукс; ЕМИАС; МСВСфера 6.3; ОСНОВА; Стрелец; Циркон 36С; СинтезМ; СинтезМ-К; AlterOS; UBLinux).
- MacOS 10.13/10.14/10.15/11/12/13/14/15 (только для класса КС1).

– Поддерживаемые виртуальные среды и гипервизоры:

- Microsoft Hyper-V Server 2008 / 2008R2 / 2012 / 2012R2 / 2016 / 2019 / 2022 (x64);
- Microsoft Hyper-V 8 / 8.1 / 10 (x64);
- Citrix XenServer 7 / 7.1 / 7.2 / 8 / 8.1 / 8.2 (x64);
- VMWare WorkStation 11 / 12 / 14 / 15 / 16 / 17 (x86, x64);
- VMWare WorkStation Player 12 / 14 / 15 / 16 (x86, x64);
- VMWare vSphere ESXi/Hypervisor 5.5 / 6.0 / 6.5 / 6.7 / 7 (x64);
- Oracle VirtualBox 5.2 / 6.1 / 7 (x86, x64);
- RHEV 4 / 4.1 / 4.2 / 4.3 / 4.4 (x64);

- ROSA Virtualization 2.0 / 2.1 (x64);
- Альт Сервер Виртуализации 9 / 9.1 / 10 / 10.1 (x64, ARM64);
- QEMU 4 / 5 / 5.1 / 5.2 / 6 / 6.1 / 6.2 / 7 / 7.1 / 7.2 (x86, x64);
- Parallels Desktop (Mac OS) 17 / 18 / 19;
- VMWare Fusion 12 / 13;
- zVirt;
- ПК СВ "Брест";
- «Диспетчер подключений виртуальных рабочих мест Термидеск» в редакции «Termidesk for Astra»;
- РЕД Виртуализация

Для указанных ОС и гипервизоров рекомендуется получать обновления по безопасности. В случае подключения СКЗИ к каналам связи выходящими за пределы контролируемой территории не допускается использование ОС, производителями которых не выпускаются обновления.

В указанных виртуальных средах и гипервизорах СКЗИ функционирует по уровню защиты КС1.

Для взаимодействия устанавливаемых на ПЭВМ программных компонентов СКЗИ и СПО АМ «Рутокен TLS» Исполнение 4 используется драйвер USB CCID (Chip/Smart Card Interface Devices) из состава ОС – универсальный драйвер для группы подобных устройств, соответствующих спецификации (стандарту) ISO/IEC 7816 (ГОСТ Р ИСО/МЭК 7816).

3. Требования по обеспечению безопасности применения СКЗИ

3.1. Требования безопасности функционирования рабочих мест

1. Использующиеся АМ должны быть инициализированы при помощи КБДЖ.468243.217. АРМ ЗКИ «РУТОКЕН TLS». Запрещается инициализировать сертифицированные АМ иными программными и аппаратными средствами, а также использовать такие АМ в работе. Запрещается подключать КБДЖ.468243.217. АРМ ЗКИ «РУТОКЕН TLS» к каналам и линиям связи.
2. Запрещается оставлять АМ «РУТОКЕН TLS» без контроля, в том числе при уходе пользователя с рабочего места.
3. Запрещается вносить какие-либо изменения в программное обеспечение СКЗИ;
4. Запрещается использовать ключи свыше срока, указанного в настоящих правилах.
5. Для библиотеки PKCS11 и модуля сетевой аутентификации должен быть обеспечен контроль целостности при помощи утилиты КБДЖ.01455. Модуль контроля целостности. Контроль целостности должен выполняться перед каждым использованием СКЗИ. Если в процессе работы программы КБДЖ.01455.Модуль контроля целостности будет обнаружено несовпадение контрольных сумм, то работа с СКЗИ запрещается, до прохождения процедуры восстановления целостности ПО.
6. Восстановление целостности ПО возможно путем удаления файлов, входящих в состав утилиты Checksum и интерфейсной библиотеки PKCS#11, и их повторной установки с эталонного носителя.
7. Программное обеспечение СКЗИ используемое на ПЭВМ (АРМ ЗКИ) должно контролироваться с помощью сертифицированного по требованиям ФСБ России

МДЗ, который производит расчет контрольных сумм объектов контроля до загрузки операционной системы. МДЗ должен вычислять контрольные суммы файлов, указанные администратором и сравнивать их с эталонными значениями (для класса КС2).

В состав модулей программного обеспечения, подлежащих проверке должны быть включены исполняемые модули операционной системы, модули ПО СКЗИ, а также драйвера ПО МДЗ. В случае обнаружении искажений объектов контроля работа с СКЗИ запрещается, и должна быть выполнена процедура восстановления целостности ПО в соответствии с п. 4.5.

8. Регламентный контроль ПО СКЗИ должен выполняться каждый раз перед включением СКЗИ, но не реже, чем один раз в сутки.
9. В случае нарушения целостности системного программного обеспечения (СПО) автоматизированного модуля (АМ) СКЗИ при работе внутренней системы контроля, пользователь должен:
   – немедленно прекратить эксплуатацию данного АМ «РУТОКЕН TLS»;
   – сообщить администратору безопасности о факте нарушения целостности внутреннего ПО.

   Далее руководствоваться пунктом 4.6.

   Эксплуатация СКЗИ «РУТОКЕН TLS» с нарушенной целостностью ПО АМ «РУТОКЕН TLS» не допускается.
10. На ПЭВМ с установленной библиотекой из состава КБДЖ.01456 должны быть выполнены следующие требования:

- должны использоваться антивирусные средства;

- должно использоваться только лицензионное ПО фирм-производителей. В случае необходимости использования иного программного обеспечения, его применение должно быть санкционировано администратором безопасности. В любом случае стороннее ПО не должно содержать средств разработки и отладки приложений, а также содержать в себе возможностей, позволяющих оказывать воздействие на функционирование ПО СКЗИ;

- необходимо регулярно устанавливать пакеты обновления безопасности (Service Packs), обновлять антивирусные базы;

- при подключении к общедоступным сетям передачи данных, необходимо исключить возможность открытия и исполнения файлов и скриптовых объектов (например, JavaScript, VBScript, ActiveX) без проведения соответствующих проверок на предмет содержания в них программных закладок и вирусов;

- должна быть установлена только одна операционная система, правом установки и настройки которой должен обладать только администратор;

- должна быть отключена возможность удаленного управления ОС;

- необходимо предусмотреть меры, максимально ограничивающие доступ к ресурсам системы (системному реестру, файлам и каталогам, временным файлам, журналам системы, файлам подкачки, кэшируемой информации), неиспользуемые протоколы, сервисы и службы рекомендуется отключить.

- средствами BIOS должна быть исключена возможность работы на ПЭВМ, если во время его начальной загрузки не проходят встроенные тесты;

- в BIOS ПЭВМ определяются установки, исключающие возможность сетевой загрузки. Не должны применяться ПЭВМ с BIOS, исключающим возможность отключения сетевой загрузки ОС;

- в настройках ПО BIOS ПЭВМ должны быть определены установки, исключающие возможность загрузки операционной системы, отличной от установленной на жестком диске (для класса КС2);

- в настройках МДЗ необходимо включить ограничение времени на его инициализацию (минимальное время от включения ПЭВМ до получения управления МДЗ) (для класса КС2);

- вход в BIOS ПЭВМ должен быть защищен паролем с длиной не менее 6 символов (для класса КС2). Смена пароля должна производиться администратором не реже, чем 1 раз в 6 месяцев;

- должно быть проведено опечатывание ПЭВМ, исключающее возможность несанкционированного изменения его аппаратной части (для класса КС2).

- ПО BIOS ПЭВМ должно соответствовать документу «Временные методические рекомендации к проведению исследований программного обеспечения BIOS по документированным возможностям». Если средства вычислительной техники на которых устанавливается СКЗИ, используется для хранения обработки и(или) передачи данных, не подлежащих обязательной защите в соответствии с законодательством Российской Федерации, то данное требование носит рекомендательный характер.

Для отключения функций телеметрии на ОС Windows 10/Server 2016 необходимо выполнить следующие действия:

1. Проверить наличие и статус сервиса DiagTrack (Панель управления -> Система и безопасность -> Администрирование -> Службы).
2. Если сервис запущен, то остановить его.
3. Удалить запись регистрации сервиса DiagTrack из реестра (Пуск -> выполнить->regedit, раздел HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services. Здесь необходимо найти и удалить папку DiagTrack).
   Удалить подготовленные к отправке данные, которые сохраняются в четырех файлах с расширением *.rbs, хранящихся в директории *%ProgramData%\Microsoft\Diagnosis*. Имена файлов для production сборок ОС – event00.rbs, event01.rbs, event10.rbs и event11.rbs. Для insider сборок ОС имена могут отличаться, поэтому необходимо удалить все файлы с расширением *.rbs. При возникновении проблем с удалением данных файлов необходимо в свойствах на вкладке «Безопасность» разрешить полный доступ к файлу, а затем удалить.
4. Остановить автоматическую (AutoLogger) ETW сессию *AutoLogger-DiagTrack-Listener*, которую DiagTrack активирует в процессе своей остановки.
5. Удалить файл, в который автоматическая (AutoLogger) ETW сессия *AutoLogger-DiagTrack-Listener* сохраняла собранные данные.
   Путь к файлу хранится в реестровой записи *AutoLogger-DiagTrack-Listener* в значении *FileName*. Конфигурации автоматических (AutoLogger) ETW сессий находятся в ключе реестра *HKLM\SYSTEM\CurrentControlSet\Control\WMI\AutoLogger*. Конфигурация целевой сессии хранится в данном ключе под записью *AutoLogger-DiagTrack-Listener*.
   В настоящее время данные сохраняются в файл *%ProgramData%\Microsoft\Diagnosis\ETLLogs\AutoLogger\AutoLogger-DiagTrack-Listener.etl*.
6. Удалить запись регистрации конфигурации автоматической (AutoLogger) ETW сессии *AutoLogger-DiagTrack-Listener* из реестра.

Данные действия необходимо выполнять после каждого кумулятивного обновления, поскольку данные обновления являются по сути полной переустановкой ОС и удаленные сервисы восстанавливаются.

3.2. Специальные требования

Для обеспечения защиты ключевой и цифровой открытой информации СКЗИ «Рутокен TLS» достаточно, чтобы канал связи был реализован в виде:

- радиоканалов GSM, GPRS, 3G\4G, WiFi, а также других современных каналов мобильной или беспроводной связи, работающих в диапазоне частот несущей выше 800 МГц с цифровой модуляцией штатного информационного сигнала;

- ВОЛС, уходящей за пределы контролируемой зоны;

- проводного канала связи с установкой в нем волоконно-оптической развязки и при условии расположения входного медиаконвертора (медь-ВОЛС) рядом с СКЗИ, а выходного медиаконвертора (ВОЛС-медь) на расстоянии не менее 1м от СКЗИ.

Использование ПЭВМ с СКЗИ «Рутокен TLS» в помещениях, где циркулирует речевая акустическая информация, содержащая сведения, составляющие государственную тайну или конфиденциального характера, без проведения его специальных исследований и специальной проверки запрещается.

В случае планирования размещения СКЗИ «Рутокен TLS» в помещениях, в которых присутствует речевая акустическая и визуальная информация, содержащая сведения, составляющие государственную тайну, и (или) установлены АС содержащая сведения, составляющие государственную тайну и (или) установлены АС и системы приема, передачи, обработки, хранения и отображения информации, содержащей сведения, составляющие государственную тайну, АС иностранного производства, входящие в состав СКЗИ «Рутокен TLS», должны быть подвергнуты проверкам по выявлению устройств, предназначенных для негласного получения информации.

4. Требования по эксплуатации СКЗИ

4.1. Общие правила эксплуатации

1. Использующиеся АМ должны быть инициализированы при помощи АРМ ЗКИ «РУТОКЕН TLS». Запрещается инициализировать сертифицированные АМ иными программными и аппаратными средствами, а также использовать такие АМ в работе. Запрещается подключать АРМ ЗКИ «РУТОКЕН TLS» к каналам и линиям связи.
2. Запрещается оставлять АМ «РУТОКЕН TLS» без контроля, в том числе при уходе пользователя с рабочего места.
3. Запрещается вносить какие-либо изменения в программное обеспечение СКЗИ;
4. Запрещается использовать ключи свыше срока, указанного в настоящих правилах.
5. Период непрерывной работы СКЗИ «РУТОКЕН TLS» не должен превышать 1 суток.
6. При эксплуатации АМ «РУТОКЕН TLS» необходимо соблюдать меры предосторожности и, в том числе, не допускать механических воздействий (падений, сотрясений, вибрации) на АМ «РУТОКЕН TLS», а также не прилагать излишних усилий при его подсоединении к порту компьютера.

Диапазон рабочих температур	От +0° до +45° С
Диапазон температур хранения	От +0° до +45° С
Допустимая относительная влажность воздуха	От 0 до 80% (без конденсата)

4.2. Требования по обеспечению целостности СКЗИ

1. Для rtpkcs11ecp и модуля сетевой аутентификации (libssl, libcrypro, rtengine) должен быть обеспечен контроль целостности при помощи утилиты КБДЖ.01455 «Модуль контроля целостности» (Checksum).
2. Контроль целостности должен выполняться перед каждым использованием СКЗИ. Если в процессе работы программы КБДЖ.01455 «Модуль контроля целостности» (Checksum) будет обнаружено несовпадение контрольных сумм, то работа с СКЗИ запрещается, до прохождения процедуры восстановления целостности ПО.
3. В состав модулей программного обеспечения, подлежащих проверке, должны быть включены исполняемые модули операционной системы, модули ПО СКЗИ, а также

драйвера ПО МДЗ. В случае обнаружении искажений объектов контроля работа с СКЗИ запрещается, и должна быть выполнена процедура восстановления целостности ПО в соответствии с подпунктом 2 пункта 4.3.

4. Регламентный контроль ПО СКЗИ должен выполняться каждый раз перед включением СКЗИ, но не реже, чем один раз в сутки.
5. Контроль целостности СПО производится вызовом функции C_EX_GetTokenInfoExtended Интерфейсной библиотеки PKCS#11. Полученная в результате выполнения команды контрольная сумма СПО сравнивается с эталонной.
6. В случае нарушения целостности системного программного обеспечения (СПО) аппаратного модуля (АМ) СКЗИ при работе внутренней системы контроля, пользователь должен:
   - немедленно прекратить эксплуатацию данного АМ «РУТОКЕН TLS»;
   - сообщить администратору безопасности о факте нарушения целостности внутреннего ПО.
   - Эксплуатация СКЗИ «РУТОКЕН TLS» с нарушенной целостностью ПО АМ «РУТОКЕН TLS» не допускается.

4.2.1. Средства контроля целостности

Для обеспечения контроля целостности модулей используется утилита контроля целостности Checksum, входящая в состав дистрибутива СКЗИ «РУТОКЕН TLS». Проверка осуществляется в соответствии с алгоритмом хэширования HMAC_GOSTR3411_2012_256 на нулевом ключе.

Утилита контроля целостности представляет собой приложение командной строки.

Перед запуском утилиты необходимо создать файл, содержащий хеш-значения контролируемых файлов (UserARM, rtPKCS11ECP, Checksum, libcrypto, libssl, rtengine) и пути к ним: в начале строки указывается предварительно вычисленное значение хэш-функции, печатается два пробела, затем указывается путь к файлу, для которого вычислялся хеш, например:

50c7ef376e75368fccafa00d197ef93bb9bcb5c0571d4ed31776f6d7401ace35
C:\\Windows\System32\rtPKCS11ECP.dll

Для выполнения проверки утилита Checksum.exe (checksum в случае ОС Linux, FreeBSD, Mac) запускается в командной строке при помощи команды

>Checksum.exe file

Где **file** – путь к файлу, содержащему значения хэш-функций и пути к файлам.

Результатом будет заново вычисленное значение хэш-функции для указанных модулей и результат его сравнения со значением, прописанным в созданном файле.

В случае использования утилиты под ОС типа *nix **file** должен формироваться редактором, поддерживающим unix-переносы строк. На таких ОС утилита имеет имя checksum. Путь в файле **file** должен прописываться с учётом программно-аппаратной платформы.

Программное обеспечение СКЗИ используемое на ПЭВМ (АРМ ЗКИ) должно контролироваться с помощью сертифицированного по требованиям ФСБ России МДЗ, который производит расчет контрольных сумм объектов контроля до загрузки операционной системы. МДЗ должен вычислять контрольные суммы файлов, указанные администратором и сравнивать их с эталонными значениями (для класса КС2).

В состав модулей программного обеспечения, подлежащих проверке должны быть включены исполняемые модули операционной системы, модули ПО СКЗИ, а также драйвера ПО МДЗ (для класса КС2).

4.3. Требования к аутентификации и разграничению доступа

Аутентификация в СКЗИ «РУТОКЕН TLS» основывается на PIN-коде пользователя, который должен состоять не менее, чем из 6 символов (до 32). Символы могут включать в себя как буквы и цифры, так и знаки препинания и т. п., т. е. любые символы, которые можно ввести со стандартной клавиатуры. Срок действия пароля до смены не должен превышать 6 месяцев.

В зависимости от введенного PIN-кода, прошедший аутентификацию пользователь получает одну из следующих ролей, различающихся полномочиями в АМ «РУТОКЕН TLS»:

- «Пользователь АМ», который имеет право менять свой PIN-код; данной ролью должны обладать конечные пользователи АМ «РУТОКЕН TLS»;
- «Администратор АМ», который имеет право менять как свой PIN-код, так и PIN-код пользователя; права администратора АМ необходимы администратору АРМ ЗКИ «РУТОКЕН TLS» для выполнения инициализации АМ «РУТОКЕН TLS», загрузки в него ключевой информации, создания хранилища сертификатов и доверенной загрузки списка сертификатов – см. документ «АРМ ЗКИ «РУТОКЕН TLS». Правила пользования».

После 3 неуспешных попыток аутентифицироваться по PIN-коду пользователя или администратора, функционирование СКЗИ блокируется. При этом если аутентификация производилась по PIN-коду пользователя, администратор с помощью своего PIN-кода может разблокировать функционирование АМ для данного пользователя.

4.4. Ввод СКЗИ в эксплуатацию

1. Рабочее место Администратора и Пользователя должны удовлетворять требованиям, изложенным в п.4.2 данного документа.
2. Администратор и Пользователь СКЗИ должны пройти соответствующую подготовку по правилам работы с СКЗИ и изучить эксплуатационную документацию на СКЗИ.
3. Для инициализации АМ «РУТОКЕН TLS» на АРМ ЗКИ, программное обеспечение АРМ ЗКИ устанавливается на ПЭВМ в соответствии с документом «АРМ ЗКИ «РУТОКЕН TLS». Правила пользования».
4. Инициализация АМ «РУТОКЕН TLS» на АРМ ЗКИ выполняется Администратором безопасности на АРМ ЗКИ «РУТОКЕН TLS». Порядок инициализации АМ «РУТОКЕН TLS» подробно описан в документе «АРМ ЗКИ «РУТОКЕН TLS». Правила пользования». Затем АМ «РУТОКЕН TLS» передается пользователю.
5. На ПЭВМ пользователя должна быть установлена библиотека rtpkcs11ecp, модуль сетевой аутентификации и утилита контроля целостности КБДЖ.01455 «Модуль контроля целостности».
6. Перед первым использованием СКЗИ требуется выполнить расчет контрольных сумм установленного ПО (программным модулем КБДЖ.01455 «Модуль контроля целостности») в соответствии с п. 6 и сравнить полученные контрольные суммы с эталонными, указанными в формуляре на СКЗИ.

Инициализацией АМ считается инициализация ключа ПДСЧ, а также выработка ключей эмитента (в случае их использования) в соответствии с АРМ ЗКИ «РУТОКЕН TLS». Правила пользования. КБДЖ.468243.217 ПП.

4.5. Действия по восстановлению работоспособности в нештатных ситуациях

Ниже приведена таблица с основным перечнем нештатных ситуаций и действиями в случае возникновения таких ситуаций.

Таблица 1

№ п/п		Нештатная ситуация	Действия при нештатной ситуации
1.		Компрометация устройства	В случае компрометации устройства необходимо следовать действиям, описанным в п. 5.5 данного документа.
2.		Выход аппаратного модуля из строя	Необходимо сообщить администратору безопасности о выходе из строя аппаратного модуля и обеспечить его доставку администратору безопасности для выяснения причин выхода из строя.
3.		Отказы и сбои в работе средств защиты от НСД.	При отказах и сбоях в работе средств защиты от НСД, администратор безопасности, должен восстановить работоспособность средств НСД. При необходимости переустановить программно-аппаратные средства НСД.
4.		Отказы в работе программных средств вследствие случайного или умышленного их повреждения.	При отказах в работе программных средств, в следствии случайного или умышленного их повреждения, лицо, ответственное за безопасность функционирования программных и аппаратных средств, обязано произвести служебное расследование по данному факту с целью установления причины отказа и восстановления правильной работы программных средств в установленном порядке.

4.6. Порядок выполнения технического обслуживания, ремонт и утилизация

В случае возникновения ситуации, требующей проведения технического обслуживания, требуется

- обратиться к представителю предприятия-изготовителя;
- уничтожить ключи пользователей, хранящиеся в АМ «Рутокен TLS» при помощи форматирования утилитой из состава «АРМ ЗКИ».
- передать неисправное устройство представителю предприятия-изготовителя;
- совместно с представителем предприятия-изготовителя определить и по возможности устранить причину отказа АМ.
- в случае невозможности устранить причину отказа АМ, следует утилизировать ключевой носитель в соответствии с п. 5.2. данного документа.

4.7. Требования к встраиванию

СКЗИ «Рутокен TLS» встраивается в прикладные системы. Встраивание должно осуществляться через функции, указанные в Приложение 1, Приложение 2.

Для встраивания СКЗИ «Рутокен TLS» в прикладные системы необходимо проводить оценку влияния аппаратных, программно-аппаратных и программных средств сети (системы) конфиденциальной связи, совместно с которыми предполагается штатное функционирование СКЗИ, на выполнение предъявленных к СКЗИ требований в случаях, определенных в подпункте 3 пункта 1 «Положения о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации (Положение ПКЗ-2005). Указанную проверку необходимо проводить по техническому заданию, согласованному с ФСБ России.

4.8. Подготовка АМ к эмиссии

Для дополнительной инициализации АМ при подготовке к эмиссии вендором реализуется прикладное ПО АРМ Инициализации, которое обеспечивает:

- генерацию закрытых ключей на АМ «Рутокен TLS»;
- выработку запросов на сертификаты;
- импорт доверенных сертификатов на АМ «Рутокен TLS»;
- импорт и обновление образов программного обеспечения во flash-памяти АМ «Рутокен TLS»;
- загрузку списка отозванных сертификатов в АМ «Рутокен TLS»;
- установку PIN пользователей.

Вышеперечисленные пункты реализуются при помощи функций СКЗИ Рутокен TLS, перечисленных в Приложении 1, Приложении 2.

АРМ Инициализации можно подключать к компьютерной сети при соблюдении мер, которые приняты в организации для работы с информацией ограниченного распространения.

Встраивание ПО АРМ инициализации должно производиться в соответствии с п. 4.7.

5. Ключевые носители СКЗИ «Рутокен TLS» Исполнение 1

5.1. Ключевые носители

Ключевыми носителями (ключевыми документами) СКЗИ являются:

- личные ключевые носители Пользователей АМ – аппаратные модули «РУТОКЕН TLS» Исполнение 1, содержащие ключи шифрования, ключи ЭП, ключи проверки ЭП, доверенные сертификаты ключей ЭП и ключ эмитента;
- ключевые носители для МДЗ (в случае его использования).

Ключевая информация на выведенных из действия аппаратных модулях «РУТОКЕН TLS» Исполнение 1 подлежит уничтожению в соответствии с п. 0.

Пароли для доступа к АМ «РУТОКЕН TLS» (PIN-коды) являются персональными паролями и должны сохраняться в тайне.

5.2. Хранение ключевых носителей

Ключевые носители пользователей должны храниться в личных сейфах или непосредственно у пользователей.

5.3. Сроки действия ключевой информации

Срок действия

а) ключа ЭП – 3 года;

б) ключа проверки ЭП – 15 лет;

в) ключ программного датчика случайных чисел (ПДСЧ) – 3 года;

г) симметричного ключа шифрования и имитозащиты – 3 года;

д) закрытого ключа обмена – 3 года;

е) открытого ключа обмена – 3 года;

Смена ключа эмитента должна осуществляться заблаговременно до окончания срока действия.

Контроль срока использования ключей должен обеспечиваться организационно-техническими мерами во избежание их использования дольше заданного срока.

5.4. Плановая смена ключевой информации

После поставки АМ «Рутокен TLS» Исполнение 1 ключ ПДСЧ и доверенные сертификаты могут использоваться 3 года. В дальнейшем их необходимо обновлять при помощи АРМ ЗКИ «РУТОКЕН TLS» не реже, чем раз в 3 года, в соответствии с КБДЖ.468243.217 ПП «АРМ ЗКИ «РУТОКЕН TLS». Правила пользования».

5.5. Компрометация ключевых носителей

Под компрометацией ключевых носителей понимают любое событие, которое влечет явную или неявную утрату доверия к тому, что используемые ключевые носители обеспечивают безопасность информации. К таким событиям относятся, включая, но не ограничиваясь, следующие:

- потеря АМ «Рутокен TLS» Исполнение 4;
- потеря ключевых носителей с их последующим обнаружением;
- появление следов механического вскрытия корпуса АМ «Рутокен TLS» Исполнение 4;
- увольнение сотрудников, имевших доступ к ключевой информации;
- назначение на должность, не связанную с использованием СКЗИ, сотрудников, имевших ранее доступ к ключевой информации;
- нарушение правил хранения ключей;
- возникновение подозрений на утечку информации;
- нарушение печати на сейфе с ключевыми носителями;
- случаи, когда нельзя достоверно установить, что произошло с АМ «Рутокен TLS» Исполнение 4, являющимся носителем ключевой информации (в том числе случаи, когда изделие вышло из строя, и доказательно не опровергнута возможность того, что данный факт произошел в результате несанкционированных действий злоумышленника).

Факты потери АМ «Рутокен TLS» Исполнение 4, вскрытия корпусов и увольнения сотрудников должны трактоваться как явная компрометация действующих ключей. Остальные события (неявная компрометация) требуют специального рассмотрения в каждом конкретном случае.

При компрометации АМ «РУТОКЕН TLS» Исполнение 1 пользователь должен немедленно прекратить работу с данным устройством и поставить в известность Администратора АМ.

По факту компрометации должно быть проведено служебное расследование с участием Администратора АМ.

После проведения служебного расследования АМ «РУТОКЕН TLS» Исполнение 1 должен быть уничтожен установленным порядком.

5.6. Уничтожение ключевой информации и ключевых носителей

Уничтожение ключей на АМ «Рутокен TLS» Исполнение 1, не выведенных из действия, возможно при помощи следующих функций интерфейсной библиотеки PKCS#11 из состава СКЗИ «Рутокен TLS» Исполнение 1 (см. Приложение 1):

- C_DestroyObject() – удаление объекта PKCS#11;
- C_InitToken() – форматирование АМ с удалением всех объектов PKCS#11.

При уничтожении ключей с помощью указанных команд должен быть составлен акт с указанием конкретного уничтоженного ключа.

Уничтожение ключей на выведенных из действия или скомпрометированных АМ «Рутокен TLS» Исполнение 1 должно производится путем их форматирования с помощью АРМ ЗКИ «Рутокен TLS» в случае его наличия в эксплуатирующей организации, или путем разрушения АМ «Рутокен TLS» Исполнение 1 в области микроконтроллера и flash-памяти с помощью кусачек или других инструментов, в случае отсутствия.

Уничтожение ключей на неработоспособных АМ «Рутокен TLS» Исполнение 1 должно производится путем разрушения АМ «Рутокен TLS» Исполнение 1 в области микроконтроллера и flash-памяти с помощью кусачек или других инструментов.

На АМ «Рутокен TLS» Исполнение 1, подлежащие уничтожению, составляется акт.

5.7. Ведение журналов операций

В данном разделе приводится рекомендуемый перечень и содержание документов для Администраторов безопасности.

Ведутся следующие журналы:

1. «Журнал регистрации администраторов безопасности», в котором фиксируются зарегистрированные администраторы безопасности, их заместители и доверенные представители.
2. «Журнал событий», где отражаются все важные события: плановая смена ключей, факты компрометации ключевых носителей, нештатные ситуации.
3. Администраторы безопасности должны также вести «Журнал пользователей», в котором записываются данные о выданных пользователю ключевых носителях (аппаратные модули «РУТОКЕН TLS») и о нештатных ситуациях, произошедших на рабочих местах с установленными СКЗИ.

Учёт СКЗИ ведется по регистрационным номерам, присваиваемым каждому экземпляру СКЗИ. Для учёта СКЗИ следует фиксировать события в следующих журналах:

1. Журнал регистрации СКЗИ. В данном журнале фиксируются все регистрационные номера используемых СКЗИ, их состав, идентификационные номера аппаратных модулей.
2. Журнал учёта ключевой информации. В данном журнале фиксируется, какие ключи записываются на аппаратный модуль.

3. Журнал назначения СКЗИ. В данном журнале указываются имена пользователей и номера соответствующих СКЗИ, выданных пользователям.

Приложение 1

Перечень вызовов функций интерфейса библиотеки rtpkcs11ecp, использование которых разрешено при встраивании СКЗИ в прикладные системы.

Таблица 2

Название	Описание	Комментарий
Функции общего назначения		
C_GetFunctionList()	Получение всех функций в библиотеке	
CK_C_EX_GetFunctionList Extended()	Получение списка функций расширений библиотеки	
C_Initialize()	Инициализация библиотеки	
C_Finalize()	Деинициализация библиотеки	
C_GetInfo()	Получение информации о библиотеке	
Функции для работы со слотами и токенами		
C_GetSlotList()	Получение списка слотов в системе	
C_GetSlotInfo()	Получение информации о слоте	
C_WaitForSlotEvent()	Ожидание события в любом слоте	
C_GetMechanismList()	Получение списка механизмов	
C_GetMechanismInfo()	Получение информации о механизме	
C_GetTokenInfo()	Получение информации об устройстве в конкретном слоте	

Название	Описание	Комментарий
C_EX_GetTokenInfo Extended()	Получение расширенной информации об устройстве	
C_InitToken()	Инициализация памяти устройства	Только Администратор
C_InitPIN()	Инициализация ПИН-кода Пользователя	Только Администратор
C_SetPIN()	Изменение ПИН-кода Пользователя	Только Администратор или Пользователь (в зависимости от настроек АМ)
C_EX_InitToken()	Расширенная инициализация памяти устройства: полностью очищает память устройства и сбрасывает все настройки и PIN-коды	Только Администратор
C_EX_UnblockUserPIN()	Разблокировка ПИН-кода Пользователя	Только Администратор
C_EX_SetTokenName()	Задание имени АМ	
C_EX_GetTokenName()	Считывание имени АМ	
C_EX_SetLicense()	Запись информации о лицензировании приложения (это массив данных, формат которых определяется разработчиком, который встраивает устройство)	Только Администратор
C_EX_GetLicense()	Считывание информации о лицензировании приложения	
C_EX_TokenManage()	Разделяет файловую структуру токена на независимые области памяти	

Название	Описание	Комментарий
Функции для работы с сессиями		
C_OpenSession()	Открытие новой сессии	
C_CloseSession()	Закрытие сессии	
C_CloseAllSessions()	Закрытие всех сессий	
C_GetSessionInfo()	Получение информации о сессии	
C_Login()	Вход Пользователя / Администратора	
C_Logout()	Выход Пользователя / Администратора	
Функции для работы с объектами		
C_CreateObject()	Создание объекта	При импорте сертификата разрешаются следующие значения поля algorithm структуры SubjectPublicKeyInfo: id-tc26-gost3410-12-256 или id-tc26-gost3410-12-512
C_ DestroyObject()	Уничтожение объекта	
C_GetAttributeValue()	Получение информации об атрибутах объекта	
C_SetAttributeValue()	Изменение информации об атрибутах объекта	
C_FindObjectsInit()	Инициализация поиска объекта	
C_FindObjects()	Поиск объекта по заданным условиям	
C_FindObjectsFinal()	Завершение поиска объекта	

Название	Описание	Комментарий
Функции для работы с ключами		
C_GenerateKeyPair()	Формирование ключевой пары	Разрешается использование следующих механизмов: `CKM_GOSTR3410_KEY_PAIR_GEN` (для генерации ключевой пары ГОСТ Р 34.10.2012 с длиной ключа 256 бит); `CKM_GOSTR3410_512_KEY_PAIR_GEN` (для генерации ключевой пары ГОСТ Р 34.10.2012 с длиной ключа 512 бит)
C_GenerateKey()	Формирование секретного ключа	Разрешается использование следующих механизмов: `CKM_GOST28147_KEY_GEN` (для генерации секретного ключа ГОСТ 28147-89);
C_DeriveKey()	Формирование сеансового симметричного ключа	Разрешается использование следующих механизмов: CKM_GOSTR3410_12_DERIVE (для алгоритма VKO GOST R 34.10-2012, 256 и 512 бит); CKM_KDF_TREE_GOSTR3411_2012_256 (для алгоритма KDF_TREE_GOSTR3411_2012_256);
C_WrapKey()	Вычисление имитовставки от ключа и его шифрование по алгоритму CryptoPro Key Wrap (из RFC 4357)	Разрешается использование следующих механизмов: CKM_GOST28147_KEY_WRAP
C_UnwrapKey()	Расшифрование ключа и проверка имитовставки по алгоритму CryptoProKey Unwrap (RFC 4357)	Разрешается использование следующих механизмов: CKM_GOST28147_KEY_WRAP

Название	Описание	Комментарий
Функции шифрования/расшифрования ***** Необходимо обеспечить защиту от повторов сообщений при шифровании информации сторонними средствами (например, введение в сообщение метки времени создания и проверки ее получателем сообщения)		
C_EncryptInit()	Инициализация шифрования	Для шифрования ключевой информации разрешается использование механизма: CKM_GOST28147_ECB (режим простой замены, алгоритм ГОСТ 28147-89). Для шифрования любой информации разрешается использование механизма: CKM_GOST28147 (режим гаммирования с обратной связью, алгоритм ГОСТ 28147-89).
C_Encrypt()	Шифрование цельного массива данных	При использовании вычисления имитовставки совместно с операцией шифрования, имитовставка формируется при помощи функций C_SignInit/C_Sign
C_EncryptUpdate()	Продолжение шифрования данных по частям	
C_EncryptFinal()	Завершение шифрования данных по частям	
C_DecryptInit()	Инициализация расшифрования	При использовании вычисления имитовставки совместно с операцией шифрования, необходимо проверять значение имитовставки, сформированной с помощью функций C_SignInit/C_Sign
C_Decrypt()	Расшифрование целиком	
C_DecryptUpdate()	Продолжение расшифрования данных по частям	
C_DecryptFinal()	Завершение расшифрования данных по частям	

Название	Описание	Комментарий
Функции хэширования		
C_DigestInit()	Инициализация хэширования	Разрешается использование следующих механизмов: CKM_GOSTR3411_12_256 (для алгоритма ГОСТ Р 34.11–2012 с длиной хэша 256 бит); CKM_GOSTR3411_12_512 (для алгоритма ГОСТ Р 34.11–2012 с длиной хэша 512 бит)
C_Digest()	Хэширование данных целиком	
C_DigestUpdate()	Продолжение хэширования по частям	
C_DigestFinal()	Завершение хэширования	
Функции вычисления имитовставки		
C_SignInit()	Инициализация вычисления имитовставки	При этом функции необходимо передать указатель на ключ алгоритма ГОСТ 28147–89 и механизм CKM_GOST28147_MAC,
C_Sign()	Вычисление имитовставки	При этом функции должен быть передан дескриптор ключа, определенный в функции *C_SignInit()*
Функции проверки имитовставки		
C_VerifyInit()	Инициализация проверки имитовставки	При этом функции необходимо передать указатель на ключ алгоритма ГОСТ 28147–89 и механизм CKM_GOST28147_MAC
C_Verify()	Проверка имитовставки	При этом функции необходимо передать дескриптор ключа, определенный в функции *C_VerifyInit()*

Название	Описание	Комментарий
Функции создания подписи		
C_SignInit()	Инициализация процесса подписи	При этом функции необходимо передать дескриптор ключа алгоритма ГОСТ Р 34.10–2012. Разрешается использование механизмов CKM_GOSTR3410_ WITHGOSTR3411_12_ 256; CKM_GOSTR3410_ WITHGOSTR3411_12_ 512
C_Sign()	Подпись данных целиком	При этом функции необходимо передать дескриптор ключа, определенный в функции *C_SignInit().*
C_SignUpdate()	Продолжение подписи данных по частям	
C_SignFinal()	Завершение подписи данных по частям	
C_EX_PKCS7Sign()	Подпись в формате PKCS#7	При этом функции необходимо передать дескриптор ключа, определенный в функции *C_SignInit()*
Функции проверки подписи		
C_VerifyInit()	Инициализация проверки подписи	При этом функции необходимо передать дескриптор ключа алгоритма ГОСТ Р 34.10–2012. Разрешается использование механизмов CKM_GOSTR3410_ WITHGOSTR3411_12_ 256; CKM_GOSTR3410_ WITHGOSTR3411_12_ 512
C_Verify()	Проверка подписи	При этом функции необходимо передать дескриптор ключа определенный в функции *C_VerifyInit()*
C_VerifyUpdate()	Продолжение проверки подписи данных, подписанных по частям	

Название	Описание	Комментарий
C_VerifyFinal()	Завершение проверки подписи данных, подписанных по частям	
C_EX_PKCS7VerifyInit()	Инициализация процесса проверки подписи переданных на вход данных в формате PKCS#7 с проверкой цепочки сертификатов	
C_EX_PKCS7Verify()	Проверка присоединенной (attached) подписи в формате PKCS#7 и возвращение данных, которые были подписаны и сертификатов подписантов	
C_EX_PKCS7Verify Update()	Проверка отсоединенной (detached) подписи в формате PKCS#7 и возвращение данных, которые были подписаны и сертификатов подписантов	
C_EX_PKCS7VerifyFinal()	Завершение процесса проверки отсоединенной (detached) подписи в формате PKCS#7 и возвращение сертификатов подписантов	
Функции вычисления HMAC		
C_SignInit()	Инициализация процесса вычисления HMAC	Поддерживаются механизмы CKM_GOSTR3411_12_256_HMAC, CKM_GOSTR3411_12_512_HMAC на ключе ГОСТ 28147-89
C_Sign()	Вычисление HMAC данных целиком	При этом функции необходимо передать дескриптор ключа, определенный в функции *C_SignInit()*

Название	Описание	Комментарий
C_SignUpdate()	Продолжение вычисления HMAC данных по частям	
C_SignFinal()	Завершение вычисления HMAC данных по частям	
Функции проверки HMAC		
C_VerifyInit()	Инициализация проверки HMAC	Поддерживаются механизм CKM_GOSTR3411_12_256_HMAC на ключе ГОСТ 28147-89
C_Verify()	Проверка HMAC	При этом функции необходимо передать дескриптор ключа определенный в функции *C_VerifyInit()*
C_VerifyUpdate()	Продолжение проверки HMAC данных, подписанных по частям	
C_VerifyFinal()	Завершение проверки HMAC данных, подписанных по частям	
Функции формирования случайных чисел		
C_GenerateRandom()	Формирование случайной последовательности	
Функции работы с сертификатами		
C_EX_GetCertificateInfoText()	Получение информации о сертификате из токена в текстовом виде	
C_EX_CreateCSR()	Создание запроса на выпуск сертификата в формате PKCS#10	

Название	Описание	Комментарий
Функции работы с flash-памятью ***** Во flash-памяти АМ «Рутокен TLS» Исполнение 1 можно хранить только открытую общедоступную информацию		
C_EX_GetDriveSize()	Получение объема flash-памяти	
C_EX_GetVolumesInfo()	Получение информации о разделах flash-памяти	
C_EX_ChangeVolume Attributes()	Изменение атрибутов разделов flash-памяти	
C_EX_FormatDrive()	Разметка flash-памяти	
C_EX_TokenManage()	Резервирует flash-память для хранения объектов инфраструктуры вендора	
C_EX_SlotManage	Выполняет загрузку и проверку содержимого образа ПО во flash-память	

Приложение 2
Основные функции интерфейса OpenSSL

Ниже приведены функции интерфейса OpenSSL, используемые разработчиками прикладного программного обеспечение при встраивании СКЗИ «РУТОКНЕН TLS».

Функции для работы со структурами криптографических сообщений CMS

Таблица 3

Название	Описание
CMS_ContentInfo_free()	Освобождение структуры *CMS_ContentInfo*
CMS_RecipientInfo_ktri_get0_algs()	
CMS_SignerInfo_cert_cmp()	Сравнивает сертификат X509 с указанным в структуре *SignerInfo*
CMS_SignerInfo_get0_signature()	Извлечь строку подписи из структуры
CMS_SignerInfo_verify()	Проверить подпись подписанных атрибутов структуры *SignerInfo*
CMS_add1_recipient_cert()	Добавляет получателя структуру *CMS_ContentInfo* типа *enveloped data* в виде структуры *KeyTransRecipientInfo*.
CMS_dataFinal()	Завершает операции и возвращает CMS
CMS_dataInit()	Получает BIO, связанный с CMS для поточной работы. Вызывается после инициализации CMS
CMS_decrypt()	Извлечь и расшифровать содержимое упакованной структуры CMS
CMS_decrypt_set1_pkey()	Расшифровывает CMS при наличии нескольких структур *recipientInfo*
CMS_encrypt()	Зашифровать и упаковать данные в CMS структуру
CMS_get0_SignerInfos()	Возвращает все структуры *CMS_SIgnerInfo*, ассоциированные со структурой *CMS-signedData*
CMS_get0_content()	Возвращает указатель на указатель *ASN1_OCTET_STRING*, который содержит внутренний контент
CMS_get0_signers()	Возвращает сертификат (сертификаты) подписи из структуры cms, ее необходимо вызывать после успешного выполнения функции *CMS_verify()*
CMS_get1_certs()	Возвращает все сертификаты в структуре *CMS_ContentInfo*
CMS_is_detached()	Возвращает информацию о том, является *CMS detached* или *attached*
CMS_set1_signers_certs()	Ищет сертификаты подписантов CMS среди переданного списка сертификатов и среди вложенных в CMS сертификатов

Название	Описание
CMS_sign()	Создаёт и возвращает структуру *SignedData CMS*, где *pkey* - структура с информацией о ключе, соответствующему сертификату
CMS_signed_add1_attr_by_OBJ()	Добавляет *signed* атрибут в полу *SignerInfo*
CMS_signed_get0_data_by_OBJ()	Получает по типу *signed* атрибута поля *SignerInfo* его значение
CMS_signed_get_attr()	Получает атрибут из *signed* поля *SignerInfo* по его порядковому номеру в этом массиве
CMS_signed_get_attr_count()	Получает количество атрибутов *signed* поля *SignerInfo*
CMS_verify()	Проверяет структуру *SignedData* из CMS

Функции для работы с контекстом и алгоритмами хеширования сообщений
Таблица 4

Название	Описание
EVP_DigestFinal_ex()	Финализирует структуру *EVP_MD_CTX*, помещая значение хеш-суммы в буфер
EVP_DigestInit_ex()	Инициализирует структуру для начала работы с определенным алгоритмом хеширования. Алгоритм задается структурой *EVP_MD*. В случае если требуется использовать специфическую реализацию данного алгоритма, например, аппаратную, реализованную в подгружаемом модуле ENGINE, нужно указать ссылку на соответствующую engine при помощи функции *ENGINE_unregister_digests*
EVP_DigestUpdate()	Хеширует данные из буфера с использованием инициализированной структуры *EVP_MD_CTX*
EVP_get_cipherbyname()	Получение указателя на структуру *EVP_CIPHER*, соответствующую указанному алгоритму
EVP_get_digestbyname()	Получает указатель на структуру *EVP_MD*, соответствующую указанному алгоритму
EVP_CIPHER_CTX_block_size()	Возвращает размер блока шифра при использовании структуры *EVP_CIPHER_CTX*
EVP_CIPHER_CTX_cleanup()	Очищает всю информацию из контекста шифрования и освобождает память, ассоциированную с ним

Название	Описание
EVP_CIPHER_CTX_ctrl()	Устанавливает параметры шифрования
EVP_CIPHER_CTX_free()	Очищает контекст, затирает ключ, вектор инициализации
EVP_CIPHER_CTX_new()	Выделяет память под структуру *EVP_CIPHER_CTX*, выполняет первоначальную инициализацию с помощью *EVP_CIPHER_CTX_init* и возвращает указатель на структуру
EVP_CIPHER_CTX_set_padding()	Устанавливает режим дополнения до целого блока
EVP_CipherFinal_ex()	Завершает операцию шифрования, возвращая результат обработки последнего блока
EVP_CipherInit()	Выполняет подготовку контекста к операциям расшифрования и зашифрования
EVP_CipherUpdate()	Обрабатывает блок данных, находящийся в буфере
EVP_MD_CTX_cleanup()	Очищает использованную структуру *EVP_MD_CTX*, освобождая память, динамически размещенную *EVP_DigestInit_ex*
EVP_MD_CTX_create()	Создает и размещает в динамической памяти структуру *EVP_MD_CTX*. Выполняет предварительную инициализацию созданной структуры
EVP_MD_CTX_destroy()	Очищает и освобождает динамически размещенную с помощью функции *EVP_MD_CTX_create* структуру *EVP_MD_CTX*
EVP_MD_CTX_free()	Очищает контекст хэширования и освобождает ассоциированное с ним пространство
EVP_MD_CTX_md()	Возвращает структуру *EVP_MD*, описывающую использованную при инициализации данной структуры *EV_MD_CTX*
EVP_MD_CTX_new()	Создает, инициализирует и возвращает ссылку на контекст хэширования
EVP_MD_CTX_reset()	Перегружает контекст хэширования. Может быть использовано для повторного использования существующего контекста
EVP_MD_CTX_create()	Выделяет память, инициализирует и возвращает контекст дайджеста

Название	Описание
EVP_MD_CTX_destroy()	Очищает контекст дайджеста, освобождает выделенное для него пространство, может вызываться только в контексте, созданном с помощью функции *EVP_MD_CTX_create()*
EVP_MD_CTX_init()	Инициализирует контекст дайджеста
EVP_MD_CTX_size()	Возвращают размер хеш-суммы в байтах для заданного алгоритма
EVP_MD_nid()	Возвращает числовой идентификатор (NID) алгоритма хэширования
EVP_MD_size()	Возвращает размер хеш-суммы в байтах для заданного алгоритма

Функции для работы с ключами и функции форматирования электронной подписи
Таблица 5

Название	Описание
EVP_PKEY_bits()	Получает количество бит в ключе
EVP_PKEY_cmp()	Сравнивает параметры и открытые ключи в двух структурах. Поскольку структура, содержащая закрытый ключ, всегда содержит и открытый, эта функция может использоваться для проверки соответствия открытого ключа закрытому
EVP_PKEY_cmp_parameters()	Сравнивает наборы параметров двух ключей
EVP_PKEY_copy_parameters()	Копирует параметры алгоритма из одной структуры в другую
EVP_PKEY_get1_EC_KEY()	Позволяет получить параметры ключа из структуры *EC_KEY*, для используемых ключей в СКЗИ ключей ГОСТ возвращает *NULL*
EVP_PKEY_free()	Уменьшает счетчик ссылок на данную структуру *EVP_PKEY* и освобождает её, если этот счетчик ссылок стал равным 0
EVP_PKEY_id()	Возвращает тип алгоритма соответствующего данному ключу
EVP_PKEY_new()	Выделяет память для структуры *EVP_PKEY*
EVP_PKEY_type()	Возвращает базовый тип *NID* типа
EVP_cleanup()	Удаляет все алгоритмы шифрования и хэширования из внутренней таблицы OpenSSL
EVP_SignFinal()	Подписывает данные, содержащиеся в контексте

Название	Описание
EVP_VerifyFinal()	Проверяет данные, содержащиеся в контексте
EVP_SignInit_ex()	Устанавливает контекст подписи для использования типа дайджеста из engine. Контекст должен быть инициализирован при помощи функции *EVP_MD_CTX_init()* до вызова этой функции
EVP_SignUpdate()	Вычисляет хеш в контексте подписи. Эту функцию можно вызывать несколько раз для одного и того же контекста для включения дополнительных данных
EVP_VerifyInit_ex()	Настраивает контекст проверки подписи на использование типа дайджеста из модуля engine. Контекст перед вызовом данной функции необходимо инициализировать с помощью функции *EVP_MD_CTX_init()*
EVP_VerifyUpdate()	Хеширует данные в контекст проверки подписи. Эту функцию можно вызывать несколько раз для одного и того же ctx для включения дополнительных данных
EVP_get_digestbynid()	Поиск структуры *EVP_MD*, описывающие алгоритмы хэширования, в таблице OpenSSL по числовому идентификатору *nid*

Функции для работы с хранилищем настроек TLS
Таблица 6

Название	Описание
SSL_CTX_add_extra_chain_cert()	Добавляет сертификат в цепочку доверия
SSL_CTX_ctrl()	Управляет настройками объектов *SSL_CTX*
SSL_CTX_flush_sessions()	Удаляет из внутреннего кэша контекста все сохраненные сессии, которые на момент времени t устарели

Название	Описание
SSL_CTX_free()	Уменьшает счетчик ссылок на контекст на единицу. Память, занимаемая контекстом, освобождается если счетчик ссылок уменьшается до нуля. Если установлена функция обратного вызова для удаления сеанса (*SSL_CTX_sess_set_remove_cb()*), этот обратный вызов будет вызываться для каждой сессии, освобождаемой из кэша сессии ctx. Это означает, что все соответствующие сессии из внешнего кэша сессий также удаляются. Если это нежелательно, пользователь должен явно сбросить обратный вызов, вызвав *SSL_CTX_sess_set_remove_cb(ctx, NULL)* до вызова *SSL_CTX_free()*
SSL_CTX_get_cert_store()	Возвращает хранилище сертификатов удостоверяющих центров, ассоциированное с контекстом. Данная функция возвращает указатель на объект хранилища, который остается ассоциирован с контекстом, что позволяет установить режимы проверки, которые можно изменить только посредством *X509_STORE API*
SSL_CTX_get_ex_data()	Возвращает дополнительные данные, ассоциированные с данным контекстом. С контекстом может быть ассоциировано несколько непрозрачных указателей, идентифицируемых индексом. Данные приложения, возвращаемые функцией *SSL_CTX_get_app_data()* соответствуют индексу 0
SSL_CTX_get_app_data()	Возвращает данные приложения, ассоциированные с контекстом
SSL_CTX_get_session_cache_mode()	Устанавливают и возвращают режим внутреннего кэша сессий
SSL_CTX_get_timeout()	Возвращает время устаревания сохраненных сессий. Если таймаут не был явно установлен функцией *SSL_CTX_set_timeout()*, используется значение по умолчанию, заданное при компиляции библиотеки
SSL_CTX_load_verify_locations()	Определяет расположение для объекта, в который размещаются сертификаты удостоверяющего центра (УЦ), используемые для проверки сертификатов другой стороны

Название	Описание
SSL_CTX_new()	Создает новый контекст TLS для установления соединения TLS. Поддерживаемые методы могут быть получены при помощи функций *TLSv1_2_client_method()* *TLSv1_2_server_method()* Запрещается использовать функции *TLS_client_method()* *TLS_server_method()* *TLSv1_client_method()* *TLSv1_server_method()* *TLSv1_1_client_method()* *TLSv1_1_server_method().*
SSL_CTX_sess_get_cache_size()	Возвращает размер внутреннего кэша сессий
SSL_CTX_sess_set_cache_size()	Устанавливает размер внутреннего кэша сессий
SSL_CTX_set_cipher_list()	Устанавливает список допустимых наборов алгоритмов шифрования. В качестве второго аргумента может передаваться только строка GOST2012-GOST8912-GOST8912
SSL_CTX_set_default_passwd_cb()	Устанавливает функцию обратного вызова для запроса пароля
SSL_CTX_set_default_verify_paths()	Устанавливает путь по умолчанию к хранилищу доверенных сертификатов УЦ, заданный при компиляции OpenSSL
SSL_CTX_set_options()	Устанавливает опции конфигурации контекста. Эти опции наследуются созданными на базе контекста соединениями
SSL_CTX_set_session_id_context()	Устанавливает контекст указанной длины, с которым сессия может быть повторно использована для указанного объекта
SSL_CTX_set_timeout()	Устанавливает время устаревания сохраненных сессий
SSL_CTX_set_verify_depth()	Устанавливает глубину проверки цепочки сертификатов
SSL_CTX_set_verify()	Устанавливает режим проверки сертификатов (для клиента — режим проверки серверных сертификатов) и функцию обратного вызова, вызываемую для проверки сертификатов

Название	Описание
SSL_CTX_use_certificate_chain_file()	Устанавливает первый из содержащихся в указанном файле формата PEM сертификат, как сертификат своего ключа, а последующие — как сертификаты промежуточных УЦ
SSL_CTX_use_certificate()	Устанавливает в контекст сертификат, переданный в виде объекта X509
SSL_CTX_use_PrivateKey()	Устанавливает в контекст закрытый ключ, находящийся в указанной структуре *EVP_PKEY*, передаваемый по указателю
SSL_CTX_add1_chain_cert()	Добавляет сертификат x509 к цепочке, связанной с текущим сертификатом контекста
SSL_CTX_get_ex_new_index()	Используется для регистрации нового индекса для использования специальных данных
SSL_CTX_set_purpose	задает purpose (назначение) на соответствие которому проверяется цепочка сертификатов в TLS
SSL_trace	предназначена для логирования сообщений TLS протокола

Функции для работы с хранилищем настроек TLS
Таблица 7

Название	Описание
SSL_do_handshake()	Инициирует рукопожатие. Для установки клиентского соединения требуется предварительный вызов *SSL_set_connet_state()*
SSL_free()	Уменьшает счетчик ссылок на объект SSL на единицу, и уничтожает его, если счетчик равен 0. Требует предварительного закрытия сессии при помощи *SSL_shutdown(), SSL_get_shutdown()*
SSL_get1_session()	Возвращает используемую сессию и увеличивает счетчик ссылок на неё
SSL_get_error()	Возвращает код результата операции ввода-вывода TLS. В качестве параметра передается значение, возвращенное функциями *SSL_read(), SSL_write(), SSL_do_handshake*(), *SSL_connect()*
SSL_get_ex_data_X509_STORE_CTX_idx()	Возвращает индекс объекта SSL в объекте *X509_STORE_CTX*

Название	Описание
SSL_get_peer_certificate()	Возвращает сертификат представленный другой стороной
SSL_get_SSL_CTX()	Возвращает объект *SSL_CTX*, создавший данный SSL
SSL_library_init()	Выполняет инициализацию внутренних таблиц библиотеки, добавляя туда все доступные алгоритмы и криптонаборы
SSL_load_error_strings()	Инициализирует подсистему диагностики ошибок
SSL_new()	Создает объект SSL
SSL_pending()	Возвращает число байт, обработанных, буферизованных, доступных внутри ssl для немедленного чтения
SSL_read()	Читает указанное число байт из указанного соединения в указанный буфер.
SSL_SESSION_free()	Уменьшает счетчик ссылок на объект сессии
SSL_session_reused()	Проверяет, была ли использована сохраненная сессия или была согласована новая
SSL_set_bio()	Устанавливает связь объектов BIO для чтения и записи
SSL_set_connect_state()	Установка для выбранного объекта клиентского режима
SSL_set_session()	Устанавливает сессию для использования в TLS соединении
SSL_set_tlsext_host_name()	Устанавливает имя сервера расширения ClientHello
SSL_shutdown()	Инициирует завершение сессии
SSL_write()	Записывает в соединение указанное число байт из указанного буфера
SSL_connect()	Инициирует проведение рукопожатия. Результат должен контролироваться функцией SSL_get_error
SSL_ctrl()	Управляет настройками объекта SSL
SSL_trace()	Предназначена для логирования сообщений TLS протокола
SSL_up_ref()	Увеличивает счетчик ссылок на существующую SLL структуру

Таблица 8

Название	Описание
TLSv1_2_client_method()	Устанавливает TLS соединение, использующее только протокол TLSv1.2
TLSv1_2_server_method()	

Функции для работы со структурой DH объектов Диффи-Хеллмана
Таблица 9

Название	Описание
DH_free()	Освобождение объектов Диффи-Хеллмана
DH_new()	Выделение объекта Диффи-Хеллмана
DH_set0_pqg()	Устанавливает значения параметров p, q. Вызов этой функции передает управление памятью значений в объект DH
DH_set_length()	Устанавливает длину параметра, связанного с объектом DH.

Функции для работы со структурой EC_KEY, хранящей ключевые параметры
Таблица 10

Название	Описание
EC_KEY_check_key()	Выполняет проверку работоспособности объекта *EC_KEY*, чтобы подтвердить его валидность.
EC_KEY_free()	Уменьшает счетчик ссылок на объект *EC_KEY*, если счетчик равен 0, освобождает память, ассоциированную с объектом
EC_KEY_generate_key()	Генерирует ключевую пару для указанного ключевого объекта. Ключевой объект должен обладать группой *EC_GROUP*, ассоциированной с ним до вызова этой функции
EC_KEY_get0_group()	Получает объект *EC_GROUP* данного ключа
EC_KEY_new_by_curve_name()	Возвращает указатель на новый созданный объект *EC_KEY* с указанным идентификатором кривой
EC_KEY_set_asn1_flag()	Устанавливает *asn1_flag* для соответствующего объекта *EC_GROUP*
EC_GROUP_get_curve_name()	Получает NID кривой
EC_get_builtin_curves()	Получает список всех предопределенных кривых

Функция выработки случайного числа

Таблица 11

Название	Описание
RAND_seed()	Функция выработки случайного числа

Функции для работы с объектами сертификатов X509

Запрещается устанавливать сертификаты с записью в поле Signature Algorithm «GOST R 34.11-94 with GOST R 34.10-2001». Перед эксплуатацией СКЗИ «Рутокен TLS» Исполнение 1 необходимо проверить списки установленных сертификатов на наличие в поле Signature Algorithm записи «GOST R 34.11-94 with GOST R 34.10-2001» и удалить сертификаты с такими записями.

Таблица 12

Название	Описание
X509_ATTRIBUTE_free()	Освобождает объект, все указатели и подобъекты в нем
X509_ATTRIBUTE_it()	
X509_check_host()	Проверяет, соответствует ли SAN или CN указанному имени хоста
X509_check_ip_asc()	Проверяет, соответствует ли указанный сертификат IP-адресу
X509_cmp()	Сравнивает два объекта X509
X509_cmp_current_time()	сравнивает ASN1_TIME в asn1_time с текущим временем
X509_CRL_free()	Освобождает объект, все указатели и подобъекты в нем
X509_CRL_get_issuer()	Получает имена эмитентов запросов сертификатов CRL
X509_CRL_get0_by_cert()	Ищет аннулированную запись по номеру сертификата
X509_CRL_up_ref()	Инкрементирует число ссылок
X509_digest()	Возвращает хеш от DER-формат объекта
X509_EXTENSION_create_by_ NID()	Создает расширение, используя указанный NID
X509_EXTENSION_create_by_ OBJ()	Создает расширение, используя указанный OBJ
X509_EXTENSION_free()	Освобождает объект, все указатели и подобъекты в нем
X509_EXTENSION_get_data()	Возвращает данные расширения
X509_free()	Освобождает объект, все указатели и подобъекты в нем
X509_get_extended_key_usage()	Возвращает значение расширения сертификата extended key usage

Название	Описание
X509_get_extension_flags()	Извлекает общую информацию о сертификате, возвращает один или несколько следующих флагов.
X509_get_issuer_name()	Получает имя издателя сертификата
X509_get_key_usage()	Возвращает значение расширения *key usage*
X509_get_subject_name()	Получает имя объекта сертификата
X509_get_X509_PUBKEY()	Получает внутренний указатель на структуру *X509_PUBKEY*, которая кодирует сертификат
X509_get0_notAfter()	Получает поле сертификата *notAfter*
X509_get0_notBefore()	Получает поле сертификата notBefore
X509_get0_pubkey()	Декодирует ключ из сертификата, не инкрементируя число ссылок на возвращаемый ключ
X509_get0_serialNumber()	Получает серийный номер сертификата
X509_NAME_add_entry_by_NID()	Добавляет поле NID
X509_NAME_get_text_by_NID()	Возвращает текст из первой записи объекта name, который соответствует NID
X509_NAME_oneline()	Печатает версию ASCII объекта *X509_NAME* в буфер
X509_NAME_cmp()	Сравнивает две структуры *X509_NAME*
X509_NAME_free()	Освобождает объект типа *X509_NAME*
X509_NAME_new()	Размещает новый объект типа *X509_NAME*
X509_PUBKEY_get0_param()	Извлекает параметры открытого ключа
X509_REQ_add_extensions()	Добавляет в запрос X509 расширения, содержащиеся в стеке
X509_REQ_add1_attr()	Добавляет атрибут в запрос
X509_REQ_free()	Освобождает структуру *X509_REQ*, размещенную с помощью *X509_REQ_new*
X509_REQ_get_extensions()	Возвращает все расширения запроса в виде стека
X509_REQ_get_X509_PUBKEY()	Получает внутренний указатель на структуру X509_PUBKEY, которая кодирует сертификат
X509_REQ_new()	Размещает в памяти структуру *X509_REQ*
X509_REQ_set_pubkey()	Устанавливает публичный ключ запроса из указанной структуры ключевой пары

Название	Описание
X509_REQ_set_subject_name()	Устанавливает поле subject запроса
X509_REQ_sign()	Подписывает запроса закрытым ключом, содержащимся в объекте ключевой пары pkey (это должен быть тот же самый объект, открытый ключ которого был установлен в качестве ключа заявки с помощью X509_REQ_set_pubkey)
X509_REVOKED_get_ext_d2i()	Ищет расширение по указанному NID и затем декодирует его
X509_STORE_CTX_get_error()	Возвращает код ошибки контекста
X509_STORE_add_cert()	Добавляет сертификат в кэш структуры *X509_STORE*
X509_STORE_CTX_get_current_cert()	Возвращает сертификат контекста, вызвавший ошибку
X509_STORE_CTX_free()	Освобождает структуру *X509_STORE_CTX*, размещенную с помощью функции *X509_STORE_CTX*_new
X509_STORE_CTX_get_ex_data()	Добавляет специфичные данные приложения в контекст
X509_STORE_CTX_get0_chain()	Возвращает внутренний указатель, использующийся в контексте, который содержит валидную цепочку
X509_STORE_CTX_get1_chain()	Возвращает валидную цепочку, если предыдущий вызов X509_verify_cert() был успешным
X509_STORE_CTX_init()	Инициализирует структуру проверки сертификата
X509_STORE_CTX_new()	Размещает динамически в памяти структуру *X509_STORE_CTX* и возвращает указатель на неё
X509_STORE_CTX_set_error()	Устанавливает код ошибки
X509_STORE_free()	Освобождает объект *X509_STORE*
X509_STORE_new()	Создает структуру *X509_STORE*
X509_up_ref()	Инкрементирует число ссылок на объект
X509_verify_cert()	Выполняет проверку сертификата с помощью ранее созданной структуры *X509_STORE_CTX*
X509_verify_cert_error_string()	Возвращает человекочитаемую строку описания ошибки при проверке
X509V3_EXT_i2d()	Кодирует специфическую структуру расширения
X509v3_get_ext_by_NID()	Получает расширение по указанному NID

Название	Описание
X509_check_host()	Проверяет, соответствует ли SAN или CN специфическому хост-имени
X509_check_ip_asc()	Проверяет, соответствует ли сертификат IPv4 или IPv6 адресу
X509_cmp()	Сравнивает два объекта X509
X509_dup()	Копирует существующий объект
X509_free()	Освобождает структуру X509
X509_get_ext_d2i()	Получает расширение и кодирует
X509_get_issuer_name()	Получает имя издателя сертификата
X509_get_notAfter()	Возвращает дату конца действия сертификата
X509_get_notBefore()	Возвращает дату начала действия сертификата
X509_get_pubkey()	Извлекает открытый ключ из сертификата в виде структуры хранения ключевой пары *EVP_PKEY*
X509_get_serialNumber()	Возвращает серийный номер сертификата
X509_get_signature_nid()	Возвращает NID, соответствующий указанному алгоритму подписи
X509_get_subject_name()	Получает имя владельца сертификата
X509_get_version()	Возвращает численное значение поля сертификата «версия»
X509_PURPOSE_add()	Позволяет добавлять кастомные объекты
X509_PURPOSE_get_count()	Получает количество зарегистрированных объектов *X509_PURPOSE*
X509_check_ca()	Проверяет, является ли переданный в качестве аргумента сертификат сертификатом УЦ
X509_get_ext()	Извлекает расширение с указанным индексом
X509_get_ext_by_NID()	Ищет расширение с заданным числовым идентификатором (NID)
X509_PURPOSE_add	Позволяет добавлять кастомные объекты X509_PURPOSE, с помощью которых потом можно проверять сертификаты и цепочки сертификатов на соответствие тому или иному назначению
X509_PURPOSE_get_count	Получает количество зарегистрированных объектов X509_PURPOSE

Служебные функции интерфейса OpenSSL

Функции для работы с абстракцией ввода-вывода BIO

Таблица 13

Название	Описание
BIO_ctrl()	Манипулирование настройками объекта BIO
BIO_free()	Удаляет объект BIO, предварительно проделав все необходимые действия по корректному завершению операций
BIO_free_all()	Удаляет всю цепочку BIO, предварительно проделав все необходимые действия по корректному завершению операций
BIO_new()	Создание объекта для чтения-записи
BIO_int_ctrl()	Контролируют операции принятия аргументов различных типов
BIO_new_CMS()	Получает потоковый ASN1 BIO
BIO_new_file()	Создание файлового BIO
BIO_new_mem_buf()	Создается буфер BIO в памяти указанной длины только для чтения
BIO_read()	Осуществляет попытку чтения указанного количества байт из объекта и помещает считанные данные в буфер
BIO_s_file()	Возвращает функцию метода файла BIO
BIO_s_mem()	Возвращает функцию метода памяти BIO
BIO_s_null()	Возвращает метод ввода-вывода, предназначенный для отбрасывания данных
BIO_s_socket()	Возвращает сокетный метод BIO. Это упаковка процедур сокета платформы.
BIO_set_fd()	Устанавливает BIO на файловый дескриптор
BIO_write()	Записывает в BIO содержимое буфера
BIO_flush()	Записывает любые внутренние буферизованные данные, в некоторых случаях используется для сигнализации конца файла, и о том, что данные больше не будут записываться
BIO_get_mem_data()	Устанавливает указатель на начало памяти, где хранятся данные BIO, и возвращает общий объем доступных данных
BIO_read_filename()	Устанавливает BIO для использования файла на чтение

Название	Описание
BIO_reset()	Перезагружает BIO в начальное состояние
BIO_write_filename()	Устанавливает BIO для использования файла на запись

Функции синхронизации, входящие в модуль engine
Таблица 14

Название	Описание
CRYPTO_THREAD_lock_frees()	Освобождает указанный lock
CRYPTO_THREAD_lock_new()	Выделяет, инициализирует и возвращает новый r/w lock
CRYPTO_THREAD_unlock()	Разблокирует предварительно заблокированный lock
CRYPTO_THREAD_write_lock()	Блокирует указанный lock
CRYPTO_set_ex_data()	Функция установки специфических прикладных данных из структуры CRYPTO_EX_DATA
CRYPTO_num_locks()	Возвращает количество параллельных вызовов, которое должны поддерживать внешние функции синхронизации

Функции для работы со структурой ANS.1
Таблица 15

Название	Описание
ASN1_BIT_STRING_free()	Освобождает объект типа ASN1_BIT_STRING
ASN1_ENUMERATED_free()	Освобождает объект типа ASN1_ENUMERATED
ASN1_ENUMERATED_get()	Возвращает значение аргумента типа ASN1_ENUMERATED
ASN1_INTEGER_cmp()	Сравнивает два значения объектов типа ASN1_INTEGER
ASN1_INTEGER_free()	Освобождает объект типа ASN1_INTEGER
ASN1_INTEGER_new()	Создает объект типа ASN1_INTEGER
ASN1_INTEGER_set_uint64()	Устанавливает параметру типа ASN1_INTEGER значения типа uint64_t
ASN1_INTEGER_to_BN()	Конвертирует ASN1_INTEGER в BIGNUM
ASN1_OBJECT_free()	Освобождает объект типа ASN1_OBJECT
ASN1_OCTET_STRING_free()	Освобождает объект типа ASN1_OCTET_STRING
ASN1_OCTET_STRING_new()	Создает объект типа ASN1_OCTET_STRING

Название	Описание
ASN1_OCTET_STRING_set()	Выставляет тип и значение аргументу типа ASN1_OCTET_STRING
ASN1_STRING_TABLE_get()	Возвращает значение аргумента типа ASN1_STRING_TABLE
ASN1_STRING_data()	Возвращает указатель на значение объекта типа ASN1_STRING
ASN1_STRING_free()	Освобождает объект типа ASN1_STRING
ASN1_STRING_get0_data()	Возвращает указатель на значение объекта типа ASN1_STRING
ASN1_STRING_get_default_mask()	Возвращает значение маски объекта типа ASN1_STRING
ASN1_STRING_length()	Возвращает длину значения объекта типа ASN1_STRING
ASN1_STRING_new()	Создает объект типа ASN1_STRING
ASN1_STRING_set()	Выставляет тип и значение аргументу типа ASN1_STRING
ASN1_STRING_set_by_NID()	Создает объект типа ASN1_STRING, используя ограничения, накладываемые на переданный ей OID
ASN1_TIME_diff()	Возвращает разницу между объектами типа *ASN1_TIME*
ASN1_TYPE_get()	Возвращает значение аргумента типа ASN1_TYPE
ASN1_TYPE_set()	Выставляет тип и значение аргументу типа ASN1_TYPE
ASN1_UTF8STRING_free()	Освобождает объект типа ASN1_UTF8STRING
ASN1_item_d2i()	Декодирует ASN1-объект
ASN1_item_free()	Освобождает объект типа ASN1_item
ASN1_item_i2d()	Кодирует ASN1-объект
ASN1_item_new()	Создает объект типа ASN1_item
ASN1_mbstring_ncopy()	Создает объект ASN1, соответствующий переданным строковым данным

Функции преобразования данных из внешнего формата во внутренний формат OpenSSL

Таблица 16

Название	Описание
d2i_ASN1_OCTET_STRING()	Декодирует структуру *ASN1_OCTET_STRING*
d2i_ASN1_TYPE()	Декодирует структуру *ASN1_TYPE*

Название	Описание
d2i_ ASN1_UTF8STRING()	Кодирует/декодирует структуру ASN1_UTF8STRING
d2i_CMS_ContentInfo()	Декодирует структуру *CMS ContentInfo*
d2i_X509_ATTRIBUTE()	Декодирует структуру *X509_ATTRIBUTE*
d2i_X509_CRL()	Декодирует *X509 CRL*
d2i_X509_EXTENSION()	Декодирует структуру *X509_EXTANSION*
d2i_X509_NAME()	Декодирует структуру X509_NAME
d2i_X509()	Декодирует массив байтов, переданный на вход, возвращает указатель на структуру X509
d2i_X509_REQ()	Декодирует запрос сертификата PKCS#10
d2i_PKCS12_bio()	Декодирует контейнер PKCS

Функции для обработки ошибок
Таблица 17

Название	Описание
ERR_clear_error()	Очищает очередь ошибок текущего треда
ERR_error_string_n()	Генерирует читаемую строку, отображающую код ошибки, помещает ее в буфер, но не более указанной длины, при необходимости усекает строку
ERR_error_string()	Генерирует читаемую строку, отображающую код ошибки, и помещает ее в буфер
ERR_get_error()	Возвращает первый код ошибки и удаляет его из очереди
ERR_peek_last_error()	Возвращает первый код ошибки и не удаляет его из очереди
ERR_GET_LIB()	Получает номер библиотеки по коду ошибки, возвращенному функцией *ERR_get_error()*
ERR_GET_REASON()	Получает код причины по коду, ошибки, возвращенному функцией *ERR_get_error()*
ERR_free_strings()	Освобождает все ранее загруженные строки ошибок

Функции преобразования данных из внутреннего формата OpenSSL во внешний формат
Таблица 18

Название	Описание
i2d_ASN1_UTF8STRING()	Кодирует структуру ASN1_UTF8STRING
i2d_ASN1_OCTET_STRING()	Кодирует структуру ASN1_OCTET_STRING
i2d_CMS_ContentInfo()	Кодирует структуру *CMS ContentInfo*
i2d_PUBKEY()	Кодирует структуру *EVP_PKEY* с использованием формата **SubjectPublicKeyInfo**
i2d_X509_EXTENSION()	Кодирует структуру *X509_EXTENTION*
i2d_X509_NAME()	Кодирует структуру *X509_NAME*
i2d_X509()	Кодирует формат DER структуры X509
i2d_X509_PUBKEY()	Кодирует *X509_PUBKEY*
i2d_X509_REQ()	Кодирует запрос на сертификат PKCS#10

Функции для анализа поля имени
Таблица 19

Название	Описание
OBJ_create()	Добавляет новый объект во внутреннюю таблицу
OBJ_nid2ln()	Конвертирует NID в long name структуры ASN1_OBJECT
OBJ_nid2obj()	Конвертирует NID в объект структуры ASN1_OBJECT
OBJ_nid2sn()	Конвертирует NID в short name структуры ASN1_OBJECT
OBJ_sn2nid()	Возвращает соответствующий NID для указанного short name
OBJ_txt2obj()	Конвертирует текстовую строку в структуру ASN1_OBJECT

Функции инициализации и начала работы с библиотекой
Таблица 20

Название	Описание
OPENSSL_config()	Чтение конфигурационного файла
OPENSSL_init_crypto()	Функция явной инициализации

Название	Описание
OPENSSL_init_ssl()	Инициализация OpenSSL
CONF_modules_free()	Освобождает память, ассоциированную с файлом конфигурации

Функции для работы со стеком
Таблица 21

Название	Описание
OPENSSL_sk_free()	Освобождает стэк объектов. Не освобождает элементы из него.
OPENSSL_sk_new_null()	Создает пустой стэк
OPENSSL_sk_num()	Получает количество объектов в стэке
OPENSSL_sk_pop_free()	Освобождает стэк и все элементы, которые в нем лежат
OPENSSL_sk_pop()	Возвращает последний объект из стэка
OPENSSL_sk_push()	Добавляет объект в стэк
OPENSSL_sk_value()	Возвращает элемент по индексу из стэка
OPENSSL_sk_find()	Ищет в стеке элемент по указателю
OPENSSL_no_config()	Отключает настройку конфигурации, при вызове до вызова OPENSSL_config конфигурация не выполняется

Функции работы с PEM-форматом
Таблица 22

Название	Описание
PEM_read_bio_DHparams()	Чтение параметров Диффи-Хеллмана в PEM
PEM_read_PUBKEY()	Чтение открытого ключа в PEM
PEM_read_bio_X509()	Чтение сертификата в PEM
PEM_write_bio_X509()	Запись сертификата в BIO

Функции работы со структурой POLICYINFO
Таблица 23

Название	Описание
POLICYINFO_free()	Освобождение структуры POLICYINFO
POLICYINFO_new()	Создание новой структуры POLICYINFO

Функции работы со структурой POLICYINFO
Таблица 24

Название	Описание
BN_bin2bn()	Конвертирует целое число в формате big-endian в BIGNUM
BN_bn2bin()	Конвертирует абсолютное значение в формат big-endian
BN_bn2hex()	Конвертирует абсолютное значение в hex
BN_cmp()	Сравнивает два числа
BN_free()	Освобождает объект большого числа
BN_new()	Выделяет память и инициализирует структуру BIGNUM
BN_num_bytes()	Возвращает размер BIGNUM
BN_set_word()	Устанавливает для BIGNUM указанное значение BN_ULONG

Функции работы со стеком
Таблица 25

Название	Описание
sk_CMS_SignerInfo_num()	Возвращает число элементов CMS_SignerInfo в стэке
sk_CMS_SignerInfo_value()	Возвращает элемент CMS_SignerInfo по индексу из стэка
sk_POLICYINFO_new_null()	Выделяет новый пустой стек без функции сравнения
sk_POLICYINFO_pop_free()	Освобождает стэк и все элементы в нем
sk_POLICYINFO_push()	Добавление элемента в стэк
sk_X509_ATTRIBUTE_free()	Освобождает стэк, но не элементы
sk_X509_ATTRIBUTE_new_null()	Выделяет новый пустой стек без функции сравнения
sk_X509_ATTRIBUTE_push()	Добавление элемента в стэк
sk_X509_CRL_new_null()	Выделяет новый пустой стек без функции сравнения
sk_X509_CRL_num()	Возвращает число элементов X509_CRL в стэке
sk_X509_CRL_pop_free()	Освобождает стэк и все элементы в нем
sk_X509_CRL_push()	Добавление элемента в стэк
sk_X509_CRL_value()	Возвращает элемент X509_CRL по индексу из стэка

Название	Описание
sk_X509_EXTENSION_new_null()	Выделяет новый пустой стек без функции сравнения
sk_X509_EXTENSION_pop_free()	Освобождает стэк и все элементы в нем
sk_X509_EXTENSION_push()	Добавление элемента в стэк
sk_X509_EXTENSION_value()	Возвращает элемент X509_EXTENSION по индексу из стэка
sk_X509_free()	Освобождает стэк, но не элементы
sk_GENERAL_NAME_num()	Возвращает число элементов GENERAL_NAME в стэке
sk_GENERAL_NAME_value()	Возвращает элемент GENERAL_NAME по индексу из стэка
sk_X509_new_null()	Выделяет новый пустой стек без функции сравнения
sk_X509_num()	Возвращает число элементов X509 в стэке
sk_X509_pop()	Возвращает и удаляет из стэка последний элемент
sk_X509_pop_free()	Освобождает стэк и все элементы в нем
sk_X509_push()	Добавление элемента в стэк
sk_X509_value()	Возвращает элемент X509 по индексу из стэка

Функции для работы с engine
Таблица 26

Название	Описание
rt_eng_init()	Создание rtengine и регистрация его в OpenSSL
rt_eng_get0_engine()	Получение экземпляра rtengine.
rt_eng_final()	Завершение rtengine и его последующее удаление из OpenSSL
rt_eng_p11_session_new()	Оборачивает pkcs11 сессию для последующего использования ее в rtengine
rt_eng_new_p11_ossl_evp_pkey()	Оборачивает хэндлы объектов ключевой пары из PKCS11 в объект EVP_PKEY OpenSSL
rt_eng_rand_set0_p11_session()	Задает сессию pkcs11, которая будет использоваться для получения случайных чисел
rt_eng_p11_session_new_null()	Создает пустую сессию
rt_eng_invalidate_p11_ossl_evp_pkey()	функция инвалидации объекта EVP_PKEY, связанного с хэндлами ключей из PKCS11

Лист регистрации изменений									
№ изм.	**Номера листов (страниц)**				**Всего листов (страниц) в докум.**	**№ докум.**	**Входящий № сопрово-дительного докум. и дата**	**Подп.**	**Дата**
	измененных	**замененных**	**новых**	**аннулированных**					