ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

ПРЕДВАРИТЕЛЬНЫЙ НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПНСТ 344— 2018

Интеллектуальные транспортные системы

АВТОМАТИЧЕСКАЯ ИДЕНТИФИКАЦИЯ ТРАНСПОРТНОГО СРЕДСТВА И ОБОРУДОВАНИЯ. ЭЛЕКТРОННАЯ РЕГИСТРАЦИЯ ИДЕНТИФИКАЦИОННЫХ ДАННЫХ ТРАНСПОРТНЫХ СРЕДСТВ

Часть 4

Безопасный обмен данными с использованием асимметричных технологий

(ISO 24534-4:2010, NEQ)

Издание официальное

Москва
Стандартинформ
2019

gost.e-ecolog.ru

Предисловие

1 РАЗРАБОТАН Обществом с ограниченной ответственностью «Научно-исследовательский институт интеллектуальных транспортных систем» (ООО «НИИ ИТС»)

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 57 «Интеллектуальные транспортные системы»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 31 декабря 2018 г. № 76-пнст

4 Настоящий стандарт разработан с учетом основных нормативных положений международного стандарта ИСО 24534-4—2010 «Автоматическая идентификация транспортного средства и оборудования. Электронная регистрационная идентификация (ERI) транспортных средств. Часть 4. Безопасный обмен данными с использованием асимметричных технологий» (ISO 24534-4:2010 «Automatic vehicle and equipment identification — Electronic registration identification (ERI) for vehicles — Part 4: Secure communications using asymmetrical techniques», NEQ)

Правила применения настоящего стандарта и проведения его мониторинга установлены в ГОСТ Р 1.16—2011 (разделы 5 и 6).

Федеральное агентство по техническому регулированию и метрологии собирает сведения о практическом применении настоящего стандарта. Данные сведения, а также замечания и предложения по содержанию стандарта можно направить не позднее чем за 4 мес до истечения срока его действия разработчику настоящего стандарта по адресу: 101990 Москва, Армянский пер., д. 9, стр. 1 и в Федеральное агентство по техническому регулированию и метрологии по адресу: 109074 Москва, Китайгородский проезд, д. 7, стр. 1.

*В случае отмены настоящего стандарта соответствующая информация будет опубликована в ежемесячном информационном указателе «Национальные стандарты», а также будет размещена на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© Стандартинформ, оформление, 2019

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения 1
2 Нормативные ссылки 1
3 Термины и определения 2
4 Сокращения 6
5 Концепция системных коммуникаций 6
5.1 Введение 6
5.2 Описание составляющих концепции системных коммуникаций 6
5.3 Службы безопасности 13
5.4 Описание архитектуры взаимосвязи 17
5.5 Интерфейсы 19
6 Требования к интерфейсу 20
6.1 Общие положения 20
6.2 Абстрактные определения транзакций 20
6.3 Интерфейсы ERT 53
Приложение А (обязательное) Модули ASN.1 56
Приложение Б (обязательное) Форма протоколов PICS 69
Приложение В (справочное) Эксплуатационные сценарии 72
Библиография 80

ПРЕДВАРИТЕЛЬНЫЙ НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Интеллектуальные транспортные системы

АВТОМАТИЧЕСКАЯ ИДЕНТИФИКАЦИЯ ТРАНСПОРТНОГО СРЕДСТВА И ОБОРУДОВАНИЯ. ЭЛЕКТРОННАЯ РЕГИСТРАЦИЯ ИДЕНТИФИКАЦИОННЫХ ДАННЫХ ТРАНСПОРТНЫХ СРЕДСТВ

Часть 4

Безопасный обмен данными с использованием асимметричных технологий

Intelligent transport systems. Automatic vehicle and equipment identification.
Electronic registration of identification data for vehicles.
Part 4. Secure communications using asymmetrical techniques

Срок действия — с 2019—06—01
до 2022—06—01

1 Область применения

Настоящий стандарт устанавливает требования к электронной регистрации идентификационных данных (ERI), основанной на идентификаторе транспортного средства (ТС) (например, для распознавания органами государственной власти), используемой в следующих случаях:

- при электронной идентификации местных и иностранных ТС органами государственной власти;
- производстве ТС, обслуживании во время эксплуатационного срока ТС и идентификации при его окончании;
- установлении срока службы (управлении жизненным циклом ТС);
- адаптировании данных ТС (например, для международных продаж);
- идентификации в целях обеспечения безопасности;
- сокращении числа совершаемых преступлений;
- при оказании коммерческих услуг.

Политика конфиденциальности и защиты данных действует в отношении информации, приведенной в настоящем стандарте. В настоящем стандарте представлено описание концепции с точек зрения бортового оборудования и оборудования дорожной инфраструктуры, необходимых для работы системы.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ПНСТ 343—2018 Интеллектуальные транспортные системы. Автоматическая идентификация транспортного средства и оборудования. Электронная регистрация идентификационных данных транспортных средств. Часть 3. Архитектура

ПНСТ 344—2018 Интеллектуальные транспортные системы. Автоматическая идентификация транспортного средства и оборудования. Электронная регистрация идентификационных данных. Часть 4. Безопасный обмен данными с использованием асимметричных технологий

ПНСТ 345—2018 Интеллектуальные транспортные системы. Автоматическая идентификация транспортного средства и оборудования. Электронная регистрация идентификационных данных. Часть 5. Безопасный обмен данными с использованием симметричных технологий

Издание официальное

Примечание — При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год. Если заменен ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1 **авторизация** (authorization): Предоставление прав, включающее обеспечение доступа на основе прав доступа.

3.2 **активная угроза** (active threat): Угроза преднамеренного несанкционированного изменения состояния системы.

3.3 **аутентификация объекта** (entity authentification): Подтверждение того, что предприятие является заявленным.

3.4 **безотказность** (non-repudiation): Ни одно из подразделений, участвующих в сообщении, не может полностью или частично наложить запрет на участие в сообщении.

3.5 **беспроводной интерфейс** (air interface): Интерфейс без проводника между бортовым оборудованием и считывателем/опросным листом, посредством которого соединение бортового оборудования (OBE) со считывателем/запросчиком осуществляется с помощью электромагнитных сигналов.

3.6 **бортовой автор ERI** (onboard ERI writer): Автор ERI, который является частью встроенного оборудования ERI.

3.7 **верификатор** (verifier): Объект, который является или представляет объект, требующий аутентифицированного удостоверения транспортного объекта.

3.8 **внешняя запись ERI** (external ERI reader); **считыватель ERI** (ERI writer): Считыватель ERI, который не является частью ее встроенного оборудования.

Примечания

1 Внешний считыватель ERI не установлен внутри или снаружи транспортного средства.

2 Есть различие между ближним (DSRC) и удаленными внешними авторами. Считывающее устройство для приближения может быть, например, PCD, с учетом [1]. Внешний считыватель ERI может быть частью придорожного оборудования, ручного оборудования или мобильного оборудования. Удаленная внешняя запись ERI может быть частью бэк-офисного оборудования (BOE).

3.9 **время жизни** (lifetime): Период времени, в течение которого существуют предмет оборудования и функции.

3.10 **встроенное оборудование ERI** (onboard ERI equipment): Оборудование, установленное внутри или снаружи транспортного средства и используемое для целей ERI.

Примечание — Встроенное оборудование ERI содержит ERT и может также содержать дополнительные устройства связи.

3.11 **встроенный считыватель ERI** (onboard ERI reader): Считыватель ERI является частью встроенного оборудования ERI.

Примечание — Встроенный считыватель ERI может быть, например, устройством с бесконтактным соединением PCD.

3.12 **вызов** (challenge): Элемент данных, выбранный случайным образом и отправленный верификатором заявителю, который используется заявителем в сочетании с секретной информацией, хранящейся заявителем, для генерации ответа, который отправляется верификатору.

Примечание — В настоящем стандарте термин «вызов» используется также в том случае, если ERT не имеет возможности шифрования и копируется без секретной информации.

3.13 **главный** (principal): Объект, чья личность может быть аутентифицирована.

3.14 **данные ERI** (ERI data): Данные идентификации транспортного средства, которые могут быть получены из ERT, состоящей из идентификатора и возможных дополнительных данных транспортного средства.

3.15 **держатель ERT** (ERT): Юридическое или физическое лицо, имеющее ERT.

Примечание — Держателем ERT может быть, например, держатель регистрационного номера или владелец, оператор или хранитель транспортного средства.

3.16 **дополнительные данные транспортного средства** (additional vehicle data): Данные электронной регистрации идентификационных данных ERI в дополнение к идентификатору транспортного средства.

3.17 **запись ERI** (ERI writer): Устройство, используемое для непосредственного или косвенного ввода данных ERI в ERT путем вызова транзакций ERI.

Примечание — Если копир ERI обменивается блоками данных протокола ERI непосредственно по каналу передачи данных с ERT, его также называют ERR. Если он обменивается данными через один или несколько узлов, только последний узел в этой последовательности называется ERR. Как следствие внешний писатель ERI может в зависимости от конфигурации на борту действовать для некоторых транспортных средств как ERR.

3.18 **защита** (security): Защита информации и данных для того, чтобы, с одной стороны, неавторизованные лица или системы не могли их читать или модифицировать, а с другой, уполномоченным лицам или системам был предоставлен доступ к ним.

3.19 **заявитель** (claimant): Объект, являющийся или представляющий собой принципал для целей аутентификации, включая функции, необходимые для участия в обмене аутентификацией от имени принципала.

3.20 **идентификация** (identification): Действие или акт установления личности.

Примечание — См. также: идентификация транспортного средства.

3.21 **ключ** (key): Последовательность символов, управляющая работой криптографического преобразования (например, шифрование, дешифрование, вычисление функции криптографической проверки, генерация подписи или проверка подписи).

3.22 **ключ общедоступного шифрования** (public encipherment key): Открытый ключ, который определяет преобразование открытого шифрования.

3.23 **ключ частной подписи** (private signature key): Закрытый ключ, определяющий преобразование частной подписи.

3.24 **контроль доступа** (access control): Предотвращение несанкционированного использования ресурса, в том числе несанкционированным образом.

3.25 **конфиденциальность** (confidentiality): Информация, не предоставляемая или не раскрываемая неавторизованным лицам, организациям или процессам.

3.26 **конфиденциальность** (privacy): Право отдельных лиц контролировать или влиять на сбор и хранение информации, относящейся к данным лицам, включая ее адресацию.

Примечание — Так как этот термин относится к праву отдельных лиц, он не может быть общепринятым, и его следует избегать, за исключением мотивации к требованию безопасности.

3.27 **криптография** (cryptography): Дисциплина, воплощающая принципы, средства и методы для преобразования данных, для того чтобы скрыть свой информационный контент, предотвратить его необнаруженную модификацию и/или несанкционированное использование.

3.28 **маскарад** (masquerade): Представлять один объект в качестве другого.

3.29 **обнаружение манипуляции** (manipulation detection): Механизм, используемый для определения того, был ли модифицирован блок данных (случайно или преднамеренно).

3.30 **односторонняя аутентификация** (unilateral authentification): Действие, позволяющее идентифицировать один объект для удостоверения личности другого, но не наоборот.

3.31 **оператор системы ERI** (ERI transaction): Устройство записи ERI, используемое для прямого или косвенного ввода данных ERI в ERT путем вызова транзакций ERI.

Примечание — Если копир ERI обменивается блоками данных протокола ERI непосредственно по каналу передачи данных с ERT, его также называют ERR. Если он обменивается данными через один или несколько узлов, только последний узел в этой последовательности называется ERR. Как следствие внешний писатель ERI в зависимости от конфигурации на борту может действовать для определенных транспортных средств как ERR.

3.32 **орган власти** (authority): Организация, зарегистрированная согласно действующему законодательству для идентификации транспортного средства с использованием электронной регистрации идентификационных данных ERI.

3.33 **от конца до конца** (end-to-end encipherment): Шифрование данных внутри или в исходной конечной системе с соответствующей дешифровкой, происходящей только внутри или в конечной системе.

3.34 **открытый текст** (cleartext): Данные, семантическое содержание которых изложено доступно.

3.35 **открытый ключ проверки** (public verification key): Открытый ключ, определяющий преобразование общественного подтверждения.

3.36 **отличительный идентификатор** (distinguishing identifier): Информация, которая однозначно отличает тип.

3.37 **пароль** (password): Конфиденциальная информация аутентификации, обычно состоящая из строки символов.

3.38 **пассивная угроза** (passive threat): Угроза несанкционированного раскрытия информации без изменения состояния системы.

3.39 **периодическое испытание транспортного средства** (periodic motor vehicle test): Обязательный периодический (например, ежегодный) тест на пригодность транспортного средства после определенного срока эксплуатации или свидетельство о прохождении такого испытания.

3.40 **повторная атака** (replay attack): Маскарад, предполагающий использование ранее переданных сообщений.

3.41 **полномочия** (credentials): Данные, передающиеся для установления личности лица.

3.42 **порядковый номер** (sequence number): Параметр времени, значение которого соответствует заданной последовательности, не повторяющейся в течение определенного периода времени.

Примечания

1 В случае высокой безопасности ERT является типом защищенного прикладного модуля SAM.

2 ERT может быть отдельным устройством или встроено в бортовое устройство, которое также предоставляет другие возможности (например, DSRC-связь).

3.43 **промежуточный центр сертификации** (intermediate certification authority): Центр сертификации, для которого сертификаты открытого ключа выдаются центром сертификации высшего уровня.

Примечание — Это определение подразумевает, что может быть только один уровень промежуточных центров сертификации.

3.44 **расшифровка дешифрования** (decipherment decryption): Разворот соответствующей обратимой шифровки.

3.45 **регистрирующий орган транспортных средств** (registration authority of transport vehicles): Орган, ответственный за регистрацию и ведение записей транспортных средств.

Примечание — Орган может предоставлять записи транспортных средств аккредитованным организациям.

3.46 **регистрирующий орган данных ERI** (registration authority of ERI data): Организация, ответственная за запись данных ERI и данных безопасности в соответствии с местным законодательством.

Примечание — Функции регистрирующего органа данных ERI могут быть такими же, как и функции регистрирующего органа для транспортных средств.

3.47 **свидетельство регистрации** (registration certificate): Документ регистрации транспортного средства (документ или смарт-карта), выданный регистрирующим органом транспортных средств, в котором зарегистрированы транспортное средство, его владелец или арендатор.

3.48 **сертификат открытого ключа** (public key certificate): Информация открытого ключа одного лица, освидетельствованная центром сертификации и, следовательно, абсолютно недоступная другим лицам.

Примечание — В настоящем стандарте сертификат открытого ключа также определяет роль объекта, для которого предоставляется информация открытого ключа, например производителя или регистрирующего органа.

3.49 **случайное число** (random number): Параметр времени, значение которого непредсказуемо.

3.50 **список контроля доступа** (access control list): Список сущностей вместе со своими правами доступа, которым разрешен доступ к ресурсу.

Пример — Модификация, воспроизведение сообщений, вставка ложных сообщений, маскировка в качестве уполномоченного лица и отказ в обслуживании.

3.51 **транспортный идентификатор** (vehicle identification): Действие или акт идентификации транспортного средства.

Примечание — Верификатор включает в себя функции, необходимые для участия в обмене аутентификацией.

3.52 **угроза** (threat): Потенциальное нарушение безопасности.

3.53 **устройство чтения ERI** (ERI reader): Устройство, используемое для прямого или косвенного считывания данных ERI из ERT путем вызова транзакций ERI.

Примечание — Если считыватель ERI обменивается блоками данных протокола ERI напрямую по каналу передачи данных с ERT, его также называют ERR. Если он обменивается данными через один или несколько узлов, только последний узел в этой последовательности называется ERR. Как следствие внешний считыватель ERI в зависимости от конфигурации на борту может выступать для определенных транспортных средств в качестве ERR.

3.54 **хеш-код** (hash-code): Строка битов, которая является выводом хеш-функции.

3.55 **хеш-функция** (hash-function): Функция, отображающая строки битов в строки фиксированной длины битов, удовлетворяющие следующим двум свойствам:

а) для данного выхода путем вычисления невозможно обнаружить вход, который сопоставляется с этим выходом;

б) для данного выхода путем вычисления невозможно обнаружить второй вход, который отображает один и тот же выход.

Примечание — Вычислительная возможность зависит от конкретных требований безопасности и среды.

3.56 **целостность данных** (data integrity): Данные, которые не изменены или уничтожены несанкционированным образом.

3.57 **центр сертификации** (certification authority): Физическое или юридическое лицо, уполномоченное создавать сертификаты открытых ключей.

3.58 **центр сертификации высшего уровня** (top-level certification authority): Сертифицирующий орган, сертификаты которого могут быть проверены, так как его общедоступный(е) ключ(и) проверки безопасности записывается(ются) как данные только для чтения в ERT до момента ее настройки или ввода в эксплуатацию.

3.59 **цифровая подпись** (digital signature, signature): Данные или криптографическое преобразование блока данных, идентифицирующие источник для получателя блока данных, а также обеспечивающие целостность блока данных и защиту от подделки, например получателем.

Примечание — См. также «криптография» (3.29).

3.60 **частный ключ** (private key): Ключ асимметричной пары ключей объекта, который должен быть использован только этим объектом.

Примечание — В случае асимметричной системы подписи частный ключ определяет преобразование подписи; при асимметричной системе шифрования частный ключ определяет преобразование дешифрования.

3.61 **частный ключ дешифрования** (private decipherment key): Закрытый ключ, определяющий приватное преобразование дешифрования.

3.62 **число ERT** (ERT number): Номер, присвоенный и записанный в ERT, который выступает в качестве уникального идентификатора ERT.

Примечание — Предполагается, что номер ERT записывается в ERT во время его изготовления и после его записи не может быть изменен.

3.63 **шифрованный текст** (chipertext): Данные, полученные с помощью шифрования, семантическое содержание которых недоступно.

3.64 **электронный регистратор чтения ERR** (electronic registration reader, ERR): Устройство, используемое для чтения или чтения/записи данных из/в электронную регистрационную метку ERT.

Примечания

1 ERR связывается напрямую, т. е. через линию данных OSI, с ERT.

2 ERR также может быть считывателем и/или записывающим устройством ERI либо выступать в качестве ретранслятора в обмене между протоколами данных ERI, ERT и считывающим устройством ERI.

3.65 **электронная регистрация идентификационных данных;** ERI (electronic registration identification): Действие или акт идентификации транспортного средства с помощью электронных средств.

3.66 **электронная регистрационная метка ERT** (electronic registration tag, ERT): Встроенное устройство ERI, содержащее данные ERI, включая соответствующие внедренные положения безопасности и один или несколько интерфейсов для доступа к этим данным.

4 Сокращения

В настоящем стандарте применены следующие сокращения:

AEI — автоматическая идентификация оборудования (automatic equipment identification);

AES — расширенный стандарт шифрования (advanced encryption standard);

ASN.1 — абстрактная синтаксическая нотация (Abstract Syntax Notation One);

AVI — автоматическая идентификация транспортных средств (automatic vehicle identification);

BOE — бэк-офисное оборудование (back office equipment);

EN — европейские нормы [Europäische Norm (German), European Standard (English)];

ENV — европейский предварительный стандарт [Europäische Norm Vorausgabe (German), European Pre-Standard (English)]

ERI — электронная регистрация идентификационных данных (electronic registration identification);

ERR — электронный регистрационный считыватель (electronic registration reader);

ERT — электронная регистрационная метка (electronic registration tag);

EU — Европейский союз (European Union);

IEC — Международная электротехническая комиссия (International Electrotechnical Commission);

ISO — Международная организация по стандартизации (International Organization for Standardization);

OBE — бортовое оборудование (on board equipment);

OSI — объединение открытых систем (Open System Intercorrection);

PICS — заявления о соответствии реализации протокола(-ов) [Protocol Implementation Conformance Statement(s)];

PIN — персональный идентификационный номер (personal identification number);

SAM — модуль приложения безопасности (secure application module);

VIN — идентификационный номер транспортного средства (vehicle identification number).

5 Концепция системных коммуникаций

5.1 Введение

В настоящем подразделе содержится введение в контекст, согласно которому данные ERI и данные безопасности могут считываться или записываться в ERT, позволяющую идентифицировать ТС, а также представлены варианты, которые могут быть использованы при фактической реализации. Нормативные требования для интерфейсов прикладного уровня приведены в разделе 6 и приложении А. Приложение Б содержит форму, определяющую ограничения фактической реализации протокола связи.

5.2 Описание составляющих концепции системных коммуникаций

5.2.1 Идентификация регистрации транспортного средства

Электронная регистрация идентификационных данных ERI — это действие или акт идентификации ТС с помощью электронных средств.

Идентификатор, используемый для идентификации ТС, называется идентификатором ТС.

Примечания

1 Предпочтительным вариантом идентификатора ТС является VIN, который назначен его изготовителем (см. [1]), кроме того, поддерживаются альтернативы (подробнее см. ПНСТ 343—2018).

2 Более подробная информация об идентификаторе ТС и данных ERI приведена в ПНСТ 343—2018.

В настоящем стандарте в качестве однозначного идентификатора отличительной черты использована комбинация уникального идентификатора ТС и уникального номера ERT.

5.2.2 Концепция системы и поддерживаемые интерфейсы

На рисунке 1 представлены интерфейсы, для которых прикладной уровень указан в настоящем стандарте.

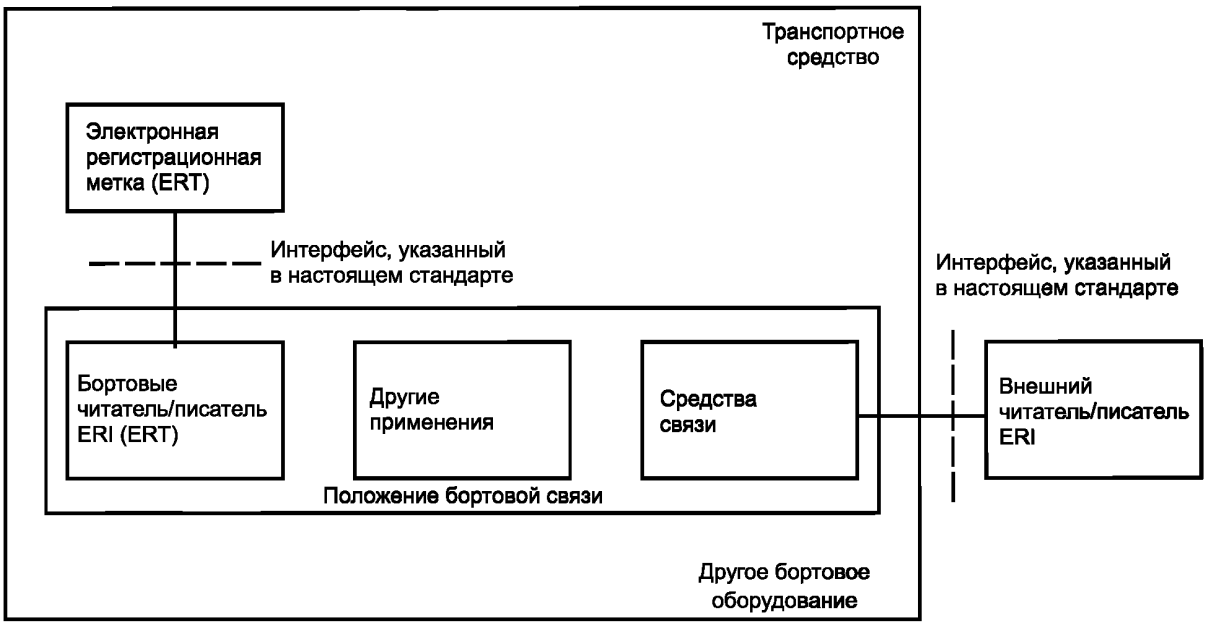

Рисунок 1 — Концепция системы и поддерживаемые интерфейсы

Встроенный компонент, обеспечивающий безопасную среду для данных ERI и данных безопасности, называется электронной регистрационной меткой ERT.

Примечание — Исполнитель может интегрировать другие положения (например, дополнительные положения о связи) в ERT при соблюдении ее безопасности.

В зависимости от своих возможностей ERT адаптируется к конкретному ТС на трех последовательных этапах (см. рисунок В.2 приложения В):

а) во-первых, ERT настроена с идентификатором ТС приложения В и, при необходимости, с дополнительными данными о ТС. Этот шаг может быть выполнен только один раз в течение жизни ERT. Пользовательская настройка не позволяет предоставлять услуги шифрования или подписи ERT;

б) во-вторых, регистрирующий орган может быть текущим регистрирующим органом ТС, добавив его данные безопасности. Регистрирующий орган может изменить свои данные о безопасности путем повторного ввода данных в действие. При поддержке ERT ввод в эксплуатацию позволяет предоставлять услуги конфиденциальности и аутентификации ERT, например необходимые ключи безопасности. Если ключ не указан, соответствующие службы не будут включены.

Примечание — Большинство смарт-карт принадлежат и контролируются одним эмитентом на протяжении всей их жизни, однако в отношении ERT ситуация иная. Когда продажа ТС осуществлена в другой стране, ERT попадает под контроль нового регистрирующего органа, поэтому будут оформлены новая номерная табличка и новый регистрационный сертификат для ТС. Затем ERT будет повторно введена в эксплуатацию;

в) в-третьих, текущий регистрирующий орган может изменить дополнительные данные ТС, для того чтобы зарегистрировать их изменение (за исключением VehicleId).

Примечание — Учитывая особенности регистрации ТС в разных странах, можно включить различные варианты приведения дополнительных данных о ТС (подробнее см. ПНСТ 343—2018).

Положения о бортовой связи должны быть способны передавать данные из/в ERT без изменения этих данных.

Примечание — Положения о бортовой связи могут быть, например, частью бортовой платформы для применения ТС.

Устройство связи может быть связано с внешним считывающим устройством или считывателем проксимити, со считывателем и/или записывающим устройством с малым радиусом действия или с удаленным бэк-офисным оборудованием (BOE).

Устройство связи, которое осуществляет связь с внешним считывающим устройством/считывателем ERI, действует как ретранслятор между этим внешним считывающим устройством/считывателем ERI и встроенным устройством чтения/записи ERI. Устройство связи также может быть использовано для других приложений.

5.2.3 Используемые роли

В настоящем стандарте выделены следующие «роли» для физических или юридических лиц:

- производители, назначающие номер VIN или шасси для каждого ТС, которое они создают. Изготовитель может также настроить ERT для конкретного ТС;
- регистрирующие органы (в отношении данных ERI), которые могут:
- назначить новое ТС (в случае дефектов) и настроить ERT (например, в случае дефектов или дооснащения).

Примечание — Регистрирующий орган может присвоить новый идентификатор ТС (например, когда номер на шасси поврежден). Затем он добавит новый идентификатор на шасси и зафиксирует его в ERT (причем идентификатор ТС не может быть перезаписан);

- поручительство себя в качестве регистрирующего органа для ТС;
- предоставление другим органам власти допуска к данным ERI;
- предоставление разрешения держателю ERT предоставлять другим поставщикам услуг доступ к данным ERI, которые отвечают за регистрацию дополнительных данных о ТС в ERT в соответствии с местным законодательством (подробнее см. ниже).

Примечания

1 Ожидается, что регистрирующий орган в отношении данных ERI является тем же органом, который хранит официальный реестр, в котором указано ТС.

2 Предполагается, что каждое ТС указано в регистре, который содержит идентификатор ТС и дополнительные данные, относящиеся к ТС. Причем этот регистр также идентифицирует ответственного за ТС (например, его владельца, оператора, хранителя, арендатора и/или водителя);

- центры сертификации, уполномоченные создавать сертификаты открытого ключа. Сертификаты открытых ключей используют для предотвращения таких ситуаций, при которых мошенническая организация может замаскироваться как производитель или регистрирующий орган. Существует два типа центров сертификации:
- центр сертификации высшего уровня и
- один промежуточный центр сертификации или более.

Примечания

1 Центр сертификации не будет напрямую связываться с ERT. Их сертификаты используют производители и регистрирующие органы.

2 С двумя уровнями центров сертификации центр высшего уровня может делегировать распространение сертификатов промежуточному центру сертификации, который отвечает за создание сертификатов для регистрирующих органов и производителей в некоторых регионах;

- органы власти, уполномоченные регистрирующим органом для считывания данных ERI с ТС (например, потому, что они имеют право делать это в силу действующего законодательства);
- дополнительные поставщики услуг (государственные или частные), которые предоставляют услугу, требующую электронной идентификации ТС и/или данных сертифицированного ТС. Владелец ERT может разрешать или не разрешать дополнительному поставщику услуг читать идентификатор ТС и дополнительные данные ТС;

- держатели ERT, которые могут быть, например, владельцем ТС, оператором или хранителем ТС.

Даже в тех случаях, когда поддерживается конфиденциальность данных ERI, владелец ERT имеет право читать ERI в своем ТС и разрешать другим поставщикам услуг читать данные идентификатора ТС. PIN-код, выданный регистрирующим органом держателю ERT, обеспечивает необходимый контроль доступа для держателя ERT.

Примечание — Роли и требования, связанные со спецификацией, проектированием и производством (включая тестирование) ERT, выходят за рамки настоящего стандарта.

5.2.4 Коммуникационный контекст для чтения

На рисунке 2 показан контекст связи для чтения данных из ERT.

Бортовой или внешний считыватель ERI используется для чтения данных из ERT. Встроенный считыватель ERI взаимодействует непосредственно с ERT и напрямую или косвенно связывается с ERT, например: непосредственно в случае карманного считывателя или интегрированного устройства ERI или опосредованно через бортовой модуль связи и встроенный считыватель ERI. Бортовой модуль связи может быть также использован для других приложений.

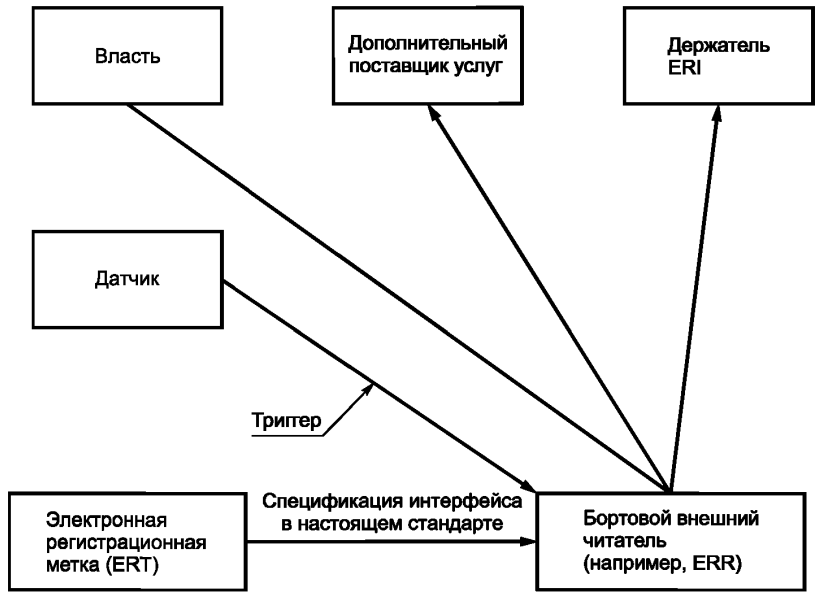

Рисунок 2 — Коммуникационный контекст для чтения из ERT

Сенсорная система (вне рамок настоящего стандарта) может быть использована для запуска внешнего считывателя ERI при фиксировании ТС, которое необходимо идентифицировать.

Лица, которые могут считывать данные ERI из ERT, представлены в 5.2.3. Права доступа различных объектов описаны в 5.3.5.1.

Владелец ERT может пожелать иметь доступ к данным ERI по различным причинам:

- для проверки правильности данных ERI;
- получения аутентифицированного (подписанного) идентификатора ТС или данных ERI, которые будут использованы для другого приложения;
- проверки списка контроля доступа (см. 5.3.5) при его наличии в ERT;
- проверки исторических данных ERI и/или исторических данных ввода в эксплуатацию при их наличии в ERT.

Оборудование, используемое органом власти, дополнительным поставщиком услуг или держателем ERT в своем офисе (т. е. в закрытом помещении), называется бэк-офисным оборудованием (BOE).

Распределение функций между BOE и внешним считывателем ERI выходит за рамки настоящего стандарта.

5.2.5 Коммуникационный контекст для написания

На рисунке 3 представлен контекст связи для записи данных в ERT.

Бортовую или внешнюю запись ERI используют для записи данных в ERT. Встроенный коммуникатор ERI напрямую связывается с ERT. Внешняя запись ERI может напрямую или косвенно связываться с ERT, например: непосредственно в случае карманного записывающего устройства либо интегрированного устройства ERI или опосредованно через бортовой модуль связи и встроенную запись ERI. Бортовой модуль связи также может быть применен для других приложений.

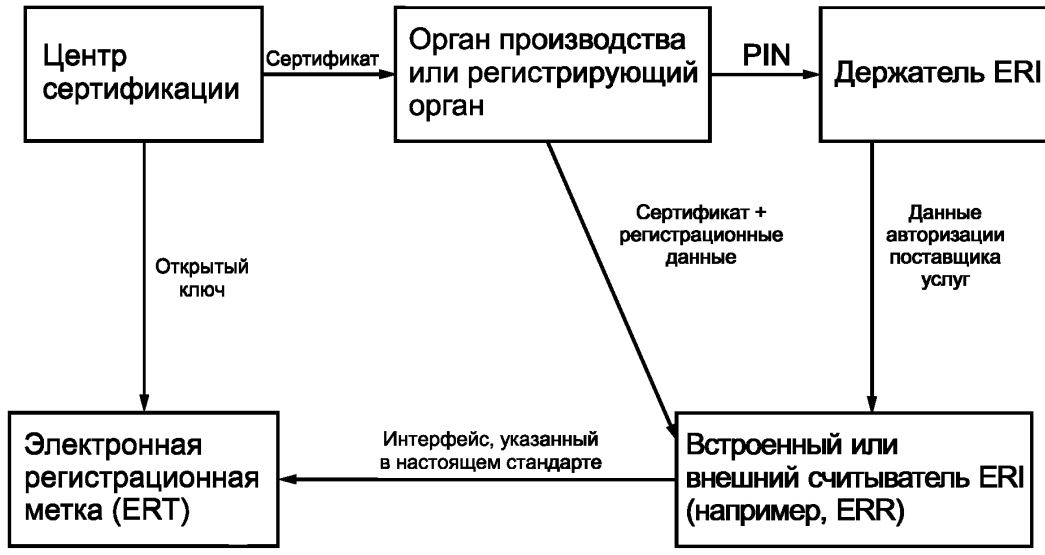

Рисунок 3 — Коммуникационный контекст для записи в ERT

Различные объекты, которые могут записывать данные ERI (безопасности) в ERT, описаны в 5.2.3. Права доступа различных объектов описаны в 5.3.5.2.

Центры сертификации предоставляют сертификаты открытого ключа для производителей и регистрирующих органов. Сертификаты используются при записи данных в ERT как доказательство подлинности (подписанных) данных, полученных от производителя или регистрирующего органа. Сертификаты также применяются для проверки подлинности данных ERI при считывании.

Регистрирующий орган наделяет полномочиями держателя ERT путем предоставления пароля (PIN). Затем владелец ERT может предоставить дополнительному поставщику услуг доступ к данным ERI.

Распределение функций между BOE и внешним устройством ERI выходит за рамки настоящего стандарта. Регистрирующий орган может поручить писателю действовать от его имени или использовать его, например: писатель является ретранслятором для удаленного письма из своего бэк-офиса.

5.2.6 Поддерживаемые уровни ERT

Предполагается, что разные пользователи могут иметь разные требования к ERI, поэтому в настоящем стандарте представлена поддержка различных уровней обслуживания возможностей ERT.

На рисунке 4 представлено описание различных уровней возможностей ERT [см. также профили соответствия протокола (PICS) в приложении Б].



Рисунок 4 — Поддерживаемые уровни ERT

Различают следующие основные варианты (вариант в наконечнике стрелок также включает в себя возможности одного из хвостов стрелки):

- только id — только чтение: это простейший вариант ERT. Идентификатор ТС записывается только один раз, и другая информация не может быть добавлена. В этом случае считыватель ERI может считывать идентификатор ТС непосредственно из ТС (ERT);

- только для чтения с дополнительными данными о ТС: в этом случае ERT может также содержать дополнительные данные о ТС (см. ПНСТ 343—2018). Однако все данные могут быть записаны только один раз в течение жизни ERT.

По соображениям безопасности данные могут также содержать подпись созданного объекта;

- чтение/запись без подтверждения аутентификации или конфиденциальности: в этом случае ERT содержит идентификатор ТС плюс дополнительные данные о ТС. При необходимости эти дополнительные данные могут быть обновлены;

- чтение/запись с аутентификацией: в этом случае ERT также предоставляет услуги аутентификации. ERT может проверять подпись, прикрепленную к записанным в нее данным. Если предусмотрено, ERT может также присоединить подпись к данным, которые она отправляет в считыватель ERI. Затем считыватель ERI может проверить, что полученные данные проистекают из подлинной ERT, а не из фальшивой.

Примечание — Регистрирующий орган может отключить возможности подписи ERT, но не ее проверки;

- чтение/запись с конфиденциальностью: в этом случае ERT может предоставлять услуги конфиденциальности. ERT может дешифровать данные безопасности, зашифрованные записью ERI. Если предусмотрено, ERT может также шифровать данные, которые отправляет в считыватель ERI. Эти данные могут быть интерпретированы только теми сторонами, которые имеют соответствующий частный ключ дешифрования.

Примечание — Регистрирующий орган может отключить возможности шифрования ERT, но не возможности дешифрования;

- чтение/запись с использованием как аутентификации, так и конфиденциальности: в этом случае ERT обеспечивает как аутентификацию, так и услуги конфиденциальности.

В дополнение к этим основным вариантам можно различать различные подварианты, например ERT:

- чтения/записи, которая также ведет журнал исторических данных, т. е. предыдущих операций записи;
- с услугами конфиденциальности, для которых регистрирующий орган может также наделить полномочиями другие органы для доступа к данным ERI;
- услугами конфиденциальности, для которых регистрирующий орган может также предоставить владельцу ERT разрешение на чтение данных (исторических) ERI;

услугами конфиденциальности, для которых регистрирующий орган может также предоставить держателю ERT разрешение на авторизацию дополнительных поставщиков услуг для чтения данных ERI.

5.2.7 Уровни безопасности и функциональная совместимость

В настоящем стандарте перечислены полномочия по идентификации ТС, включая роуминг из юрисдикции других регистрирующих органов. Поэтому большое внимание уделяется взаимодействию ERT и считывателей различных уровней возможностей, т. е. идентификация ТС посредством простого ERT с помощью сложного считывателя, и наоборот.

Основным принципом взаимодействия ERT и считывателей ERI является то, что читатель будет функционировать с полным спектром возможностей ERT. Если читатель не обеспечивает полную функциональность, например требуемые возможности дешифрования, тогда полномочия, использующие читателя, могут запрашивать помощь (дешифровочную) от регистрирующего органа. Аналогичная ситуация имеет место в том случае, когда ERT подписывает данные.

В таблице 1 представлены различные комбинации, когда ERT и считыватель ERI имеют разные криптографические возможности.

Таблица 1 — Взаимодействие между ERT и считывателями ERI

ERT	Читатель ERI	Совместимость
Шифрование:	Расшифровка:	
да	нет	Читатель не поймет данные ERI, которые нуждаются в помощи регистрирующего органа ТС для дешифрования прочитанных данных
нет	да	Читателю не нужно использовать возможности расшифровки
Подписание:	Проверка подписи:	
да	нет	Читатель не сможет проверить любую подпись, предоставленную для данных ERI.
нет	да	Читателю не нужно использовать возможности проверки подписи

Для того чтобы поддерживать читателей без возможности расшифровки, регистрирующий орган может, при необходимости, отключить возможности шифрования ERT для ТС, находящихся в его юрисдикции.

Пример — Когда ТС импортируется из страны с высоким уровнем конфиденциальности в страну с менее строгим уровнем защиты конфиденциальности (следовательно, использованы читатели, не имеющие возможности расшифровки), тогда регистрирующие органы могут отключить возможности дешифрования ERT при ее вводе в эксплуатацию.

Примечание — Согласование требований безопасности со стороны регистрирующего органа выходит за рамки настоящего стандарта.

Что касается функциональной совместимости между ERT и писателями ERI, приоритет отдается безопасности выше уровня поддержки международных (повторных) продаж. Так как регистрирующий орган может заменить ERT, это не считается серьезным ограничением. Основным принципом написания является то, что автор и, следовательно, регистрирующий орган могут определять возможности ERT, и на основании этого регистрирующий орган может решить принять данную ERT или заменить ее на соответствующую предъявляемым требованиям.

Примечание — Если регистрирующий орган не принимает определенную ERT, то регистрирующий орган должен заменить ее на приемлемую и осуществить настройку в дальнейшем по мере необходимости.

Взаимодействие ERT и писателей ERI с различными уровнями возможностей показано в таблице 2.

Таблица 2 — Взаимодействие между ERT и авторами ERI

Писатель ERI	ERT	Совместимость
Шифрование:	Расшифровка:	
да	нет	Писатель ERI может записывать данные только в ERT, если он отключает возможности шифрования
нет	да	ERT не будет принимать данные, которые должны быть зашифрованы таким автором
Подписание:	Проверка подписи:	
да	нет	ERT не будет проверять подпись, предоставленную автором. Как следствие каждый автор может быть использован для изменения данных ERI и данных безопасности. Примечание — Не рекомендована, но разрешена в рамках настоящего стандарта. Ответственный орган разрешает или не разрешает (т. е. поручать или не вводить комиссию) такого рода ERT.
нет	да	ERT не принимает данные от этого автора

Ответственный регистрирующий орган должен принять или не принять конкретную ERT в пределах своей юрисдикции, т. е. регистрирующий орган может принять решение относительно роли поручителя в конкретной ERT.

5.3 Службы безопасности

5.3.1 Предположения

Концепция безопасности обмена данными между ERT и считывателем или автором ERI основана на следующих предположениях:

- использование ERT может быть обязательным и, следовательно, должно быть устойчивым к мошенничеству. ERT (см. [2]) должна быть устойчивой к активным угрозам (например, изменение, воспроизведение сообщений, вставка ложных сообщений и маскировка);
- чтение ERT должно быть пригодным в качестве юридического доказательства (неотказуемости).

Примечание — Для этого может потребоваться спецификация профиля защиты (включая достаточный уровень обеспечения оценки, EAL) для ERT (см. [3]). Однако такая спецификация выходит за рамки настоящего стандарта;

- ERI должна иметь возможность обеспечить высокий уровень защиты частной жизни (т. е. возможности легко отслеживать модели мобильности ТС, и, следовательно, его обычного водителя не должно быть). Как следствие ERT также должна быть устойчивой к пассивным угрозам;
- ERI должна иметь возможность предусматривать меры защиты, для того чтобы предотвратить использование ERI для запуска атаки на ТС;
- эффективность механизмов безопасности должна быть достижимой в течение времени, доступного при движении ТС.

Пример — Чтение данных ТС со скоростью 180 км/ч в пределах 10-метрового диапазона считывания должно быть достигнуто в течение 200 мс.

5.3.2 Аутентификация объекта при чтении данных ERI

Доверие к подлинности чтения данных ERI зависит от следующих аспектов аутентификации, которые должны быть выполнены для того, чтобы полностью доверять чтению:

а) ERT настраивается с достоверным идентификатором ТС и прикрепляется к ТС;

б) ERT не может быть удалена из ТС, не делая его неработоспособным;

в) данные ERI считываются из подлинной ERT, т. е. из действующего на законных основаниях устройства (это нереплицированное сообщение от поддельной ERT). Это достигается путем подписания данных ERI вместе с кодом вызова, предоставляемым считывателем ERI;

г) данные ERI правильно считываются из ERT (целостность данных, обнаружение манипуляций). Это достигается с помощью стандартных механизмов, используемых в обмене данными, путем подписи данных и, в качестве побочного эффекта, шифрования данных ERI (дешифрование поврежденного зашифрованного текста не продуктивно);

д) если данные ERI правильно прочитаны из подлинной ERT по конкретному запросу, сложно впоследствии будет оспаривать, что эти данные не прочитаны из этого компонента по этому запросу (неотказуемость), что достигается путем подписания данных ERI вместе с кодом вызова, предоставляемым считывателем ERI.

Примечания

1 Пункт 5.3.2 касается только перечислений г)—д). Пункт б) указан в ПНСТ 342—2018.

2 Использование терминологии, приведенной в перечислениях г) и д) [4], поддерживается путем применения двухстороннего одностороннего механизма аутентификации. Единственность/своевременность контролируется путем генерирования и проверки случайного числа, отправленного считывателем ERI (верификатором) в ERT (заявитель).

5.3.3 Конфиденциальность при чтении данных ERI

В настоящем стандарте поддерживается конфиденциальность, предоставляя данные ERI в зашифрованном виде. Зашифрованные данные ERI могут находиться в свободном доступе, но будут расшифрованы и интерпретированы только уполномоченными лицами/оборудованием (сквозная шифровка).

Для того тобы запретить использование зашифрованных данных ERI в качестве псевдонима, к данным ERI перед шифрованием добавляется порядковый номер.

Примечание — Конфиденциальность в технических терминах обеспечивается с помощью асимметричных ключей. В случае операций чтения ERT использует общедоступный ключ шифрования органа, зарегистрированного в ERT регистрирующим органом или владельцем ERT (для дополнительного поставщика услуг). В случае операций записи конфиденциальность достигается с помощью общедоступного ключа шифрования, который может быть получен из ERT. Затем данные, которые нужно записать, могут быть дешифрованы только с помощью секретного ключа дешифрования ERT.

5.3.4 Ключи для аутентификации и конфиденциальности

Транспортное средство может быть зарегистрировано на протяжении длительного промежутка времени, так же как и многие другие ТС. Как следствие регистрирующий орган должен всегда использовать либо одни и те же ключи, либо разные ключи для разных ТС. Для последнего случая ключ можно идентифицировать с помощью идентификатора ключа.

В случае шифрования идентификатор ключа общедоступного шифрования прикрепляется к зашифрованному тексту и затем может быть использован получателем для определения соответствующего частного ключа дешифрования.

Для проверки подписи к подписанным данным добавляется один или два сертификата открытого ключа. Для данных, подписанных ERT, сертификат предоставляется регистрирующим органом в тот момент, когда секретный ключ подписи фиксируется в ERT (см. транзакцию комиссии ERT). Для данных, которые должны быть подписаны BOE, секретный ключ подписи для сертификата высшего уровня определяется с помощью ключевого идентификатора, который может быть получен из ERT. Этот ключевой идентификатор и открытый ключ проверки записываются в ERT при его изготовлении.

В соответствии с настоящим стандартом регистрирующий орган изменяет свой общедоступный ключ шифрования, ключ частной подписи и PIN-код для держателя ERT. Это может быть, например, частью периодического испытания ТС.

Примечание — Например, сотрудники гаража, выполняющие периодический тест, самостоятельно изменяют этот ключ. Несмотря на то что в рамках настоящего стандарта определены меры безопасности, которые должны применяться регистрирующими органами для защиты своих секретных ключей, более вероятно, что данные сотрудники установят связь между ERT и регистрирующим органом, который может затем безопасно обновить ERT.

5.3.5 Контроль доступа к данным ERI

5.3.5.1 Контроль доступа к чтению

а) Объекты с доступом чтения

Следующие три типа объектов могут потребовать доступа к данным ERI:

- органы власти;
- авторизованные поставщики услуг;

- держатель ERT.

Если ERT использует возможности шифрования, она контролирует доступ для чтения полномочий и уполномоченных дополнительных поставщиков услуг посредством списка контроля доступа, который содержит для каждого объекта следующие данные:

- идентификатор;
- общедоступный ключ шифрования и идентификатор ключа;
- указание на полномочия данного лица.

Открытый ключ шифрования использован для шифрования случайного секретного ключа, данных ERI, отправленных считывателю ERI.

Если регистрирующий орган не поддерживает конфиденциальность, его ключ общедоступного шифрования будет иметь нулевое значение, и любое содержимое списка управления доступом будет проигнорировано. В этом случае данные ERI будут отправляться в виде открытого текста в считыватель ERI.

б) Доступ для чтения для уполномоченных органов

Если ERT использует возможности шифрования, регистрирующий орган может разрешить полномочия для считывания данных ERI путем записи ее идентификатора, ключа общедоступного шифрования и идентификатора ключа в список управления доступом ERT (см. 5.3.5.2).

Читатель ERI уполномоченного органа может иметь конфиденциальный доступ к данным ERI, предоставляя идентификатор этого органа. Затем ERT зашифрует данные ERI с помощью своего общедоступного ключа шифрования.

в) Доступ для несанкционированного чтения органами власти

Орган власти, которому не разрешено считывать данные ERI, не должен указывать идентификатор объекта в его запросе на чтение. Затем ERT шифрует данные ERI с помощью общедоступного ключа шифрования регистрирующего органа ТС.

Затем для дешифрования данных ERI требуется частный ключ дешифрования в регистрирующем органе ТС, что может быть выполнено несколькими способами (которые не входят в рамки настоящего стандарта):

- считыватель ERI можно ввести в эксплуатацию, предоставив секретный ключ дешифрования регистрирующего органа, например SAM, и в этом случае считыватель может дешифровать данные ERI;
- локальный (регистрирующий) орган может иметь этот закрытый ключ и затем расшифровать данные ERI, полученные читателем;
- регистрирующему органу ТС может быть предложено расшифровать данные ERI.

Примечание — Процедуры сотрудничества между регистрирующими органами выходят за рамки настоящего стандарта.

г) Доступ для чтения, предоставляемый авторизованным поставщикам услуг

Доступ к данным поставщика услуг контролируется путем добавления его идентификатора, открытого ключа шифрования и идентификатора ключа в список управления доступом в ERT. Затем считыватель этого поставщика услуг может быть введен в эксплуатацию с помощью соответствующего частного ключа дешифрования.

При запросе данных ERI от ТС считыватель предоставляет ERT идентификатор поставщика услуг. Если в списке управления доступом содержится этот идентификатор, соответствующий ключ общедоступного шифрования используется для шифрования данных ERI, возвращаемых считывателю. В противном случае используется общедоступный ключ шифрования в регистрирующем органе ТС.

д) Доступ для чтения для держателя ERT

Настоящий стандарт обеспечивает доступ к данным ERI открытого текста, если запрос содержит как идентификатор vehicleId, так и пароль (PIN), поставляемый держателю ERT.

Использование только vehicleId нарушает требование о том, что ERI не будет легко использоваться для запуска атаки на ТС. Успех попытки считывания данных с использованием только VehicleId в качестве информации относительно аутентификации может затем применяться в качестве такого триггера. Использование PIN-кода снижает этот риск.

5.3.5.2 Контроль доступа к записи

а) Объекты с доступом для записи

Доступ к записи может потребоваться следующим типам объектов:

- изготовителям и регистрирующим органам для настройки ERT для конкретного ТС;

- регистрирующим органам для ввода в эксплуатацию и обновления данных ERI или данных безопасности;
- владельцу ERT.

б) Доступ к записи для настройки производителями и регистрирующими органами

Настоящий стандарт поддерживает настройку ERT для конкретного ТС производителем или регистрирующим органом.

Настройка ERT для конкретного ТС осуществляется путем написания идентификатора ТС и в конечном счете внесения дополнительных данных ТС (например, даты регистрации, регистрационного номера, типа двигателя и т. д.) в этот компонент.

ERT можно настраивать только один раз в течение срока ее эксплуатации, а идентификатор ТС в ERT не может быть изменен ни при каких обстоятельствах.

Примечание — Это может быть выполнено, например, изготовителем во время сборки ТС регистрирующим органом, при необходимости, при замене ERT (например, из-за дефекта) или ее модификации.

Для того чтобы получить доступ к записи, пишущее оборудование должно доказать, что оно действует от имени производителя или регистрирующего органа. С этой целью пишущее оборудование должно предоставить один из двух сертификатов открытого ключа:

- сертификат открытого ключа, выданный либо центром сертификации высшего уровня, либо промежуточным центром сертификации;
- если сертификат открытого ключа выдается промежуточным центром сертификации, то сертификатом открытого ключа считается сертификат, выданный центром сертификации высшего уровня.

Примечания

1 По мере того как открытый(ые) ключ(и) общественного контроля центра сертификации высшего уровня помещается в ERT при изготовлении, этот компонент может проверить подпись в сертификате, выдаваемом центром сертификации высшего уровня.

2 Так как требуется, чтобы данные, которые должны быть записаны в ERT, были выданы официальным регистрирующим органом (и содержать действительный идентификатор ТС), отсутствует необходимость санкционировать пишущее оборудование. Канал связи, используемый для обмена данными ERI от BOE регистрирующего органа в ERT, может быть небезопасным.

в) Дополнительный доступ для записи в регистрирующие органы

В дополнение к настройке настоящий стандарт также поддерживает, насколько это применимо, доступ к записи в регистрирующий орган в ERT, для того чтобы:

- поручить себя в качестве регистрирующего органа ТС;
- добавить или изменить ключ частной подписи, который будет использоваться ERT для подписания сообщений и сертификата для соответствующего открытого ключа проверки, подписанного самим пользователем.

Примечание — Этот сертификат также включает идентификатор ТС;

- добавить или изменить свой общедоступный ключ шифрования, необходимый для шифрования данных ERI (при сохранении конфиденциальности);
- разрешить другим органам, добавляя их идентификатор, использовать ключ общедоступного шифрования и идентификатор ключа в списке контроля доступа ERT (если необходимо поддерживать конфиденциальность);
- добавить или изменить PIN-код, выданный держателю ERT (при сохранении конфиденциальности).

Примечание — Каким образом и когда регистрирующий орган посылает держателю ERT (новый) PIN-код, выходит за рамки настоящего стандарта;

- изменить данные о дополнительном ТС (например, его регистрационный номер и цвет, тип двигателя и т. д.) согласно требованиям или разрешениям местного законодательства.

Доступ к записи получается путем передачи одного или двух сертификатов открытого ключа, как описано в предыдущем разделе.

г) Доступ к записи для держателей ERT

В настоящем пункте поддерживается доступ к записи держателя ERT с целью авторизации доступа для чтения:

- для дополнительного поставщика услуг путем записи идентификатора, общедоступного ключа шифрования и идентификатора ключа этого поставщика услуг в ERT;
- бортовые приложения/оборудование, которые должны проверять или аутентифицировать данные ERI, записывая идентификатор, общедоступный ключ шифрования и идентификатор ключа в ERT.

Для того чтобы получить доступ к записи, владелец ERT должен идентифицировать себя как держателя ERT, содержащего данные ERI, и с этой целью использует идентификатор ТС и пароль (PIN), полученный от регистрирующего органа.

Примечание — Каким образом и когда владелец ERT получает этот пароль (PIN), выходит за рамки настоящего стандарта.

Использование исключительно идентификатора ТС нарушает требование о том, что ERI не будет легко использоваться для запуска атаки на ТС. Принятие попытки записи данных с применением исключительно идентификатора ТС в качестве информации аутентификации может быть использовано в качестве такого триггера. Применение PIN-кода снижает этот риск.

5.3.6 Идентификация ключа государственной сертификации

Центр сертификации высшего уровня может использовать разные ключи частной подписи. Идентификатор ключа применяется для определения ключа частной подписи, который соответствует общедоступному ключу проверки, хранящемуся в ERT с правом чтения.

5.4 Описание архитектуры взаимосвязи

5.4.1 Идентификация транспортного средства с авторизованным считывателем короткого замыкания

На рисунке 5 представлена концепция взаимосвязи для идентификации ТС с авторизованным считывателем ERI с использованием коротких сообщений.

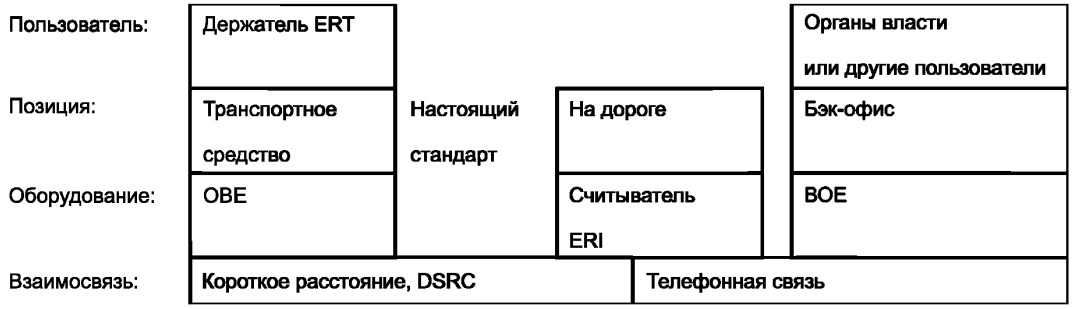

Рисунок 5 — Взаимосвязь в ближней зоне с авторизованным считывателем

В настоящем стандарте использован прикладной уровень беспроводного интерфейса между бортовым оборудованием ERI и внешними считывателями ERI с малой дальностью.

Примечание — Внешний интерфейс считывателя ERI ТС соответствует эталонной точке DELTA (см. приложение А с учетом [5]; 5.5.1). Внешний интерфейс считывателя ERI бэк-офиса соответствует эталонной точке ALPHA (см. [5]).

Описание интерфейса между внешним считывателем ERI и BOE выходит за рамки настоящего стандарта. Это может использоваться, например, для ввода в эксплуатацию считывателя ERI, обмена белыми или черными списками и/или для загрузки результатов чтения, т. е. локальным интерфейсом в бэк-офисе или в широкополосном сетевом интерфейсе.

Бэк-офис может быть офисом регистрирующего органа или дополнительного поставщика услуг.

Аналогичная ситуация относится и к аналогичной фигуре для идентификации ТС с авторизованным внешним считывателем бесконтактного доступа.

5.4.2 Идентификация транспортного средства с помощью авторизованного считывателя с малой дальностью

На рисунке 6 представлена концепция взаимосвязи для идентификации ТС с неавторизованным считывателем ERI ближнего действия, т. е. с внешним считывающим устройством ERI, которое не имеет частного ключа дешифрования для дешифрования данных ERI.

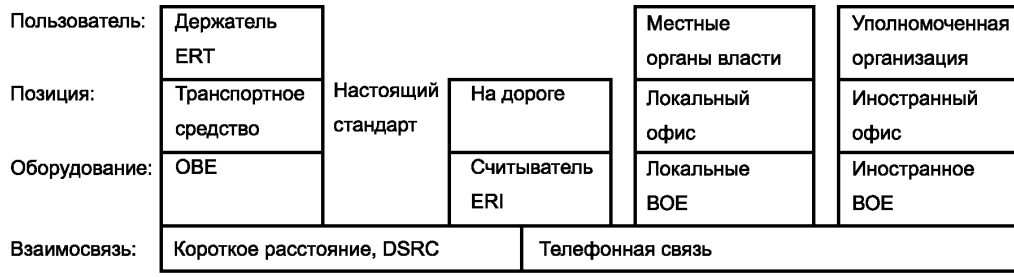

Рисунок 6 — Коммуникации с малой дальностью связи с неавторизованным считывателем

Для идентификации иностранного ТС, т. е. ТС, для которого внешний считыватель ERI не разрешен, необходимо сотрудничество с уполномоченным органом, например регистрирующим органом, который зарегистрировал ТС.

Кроме того, для получения необходимой информации можно обратиться с запросом к представителям зарубежного банка или архивировать его через собственный банк.

Аналогичная ситуация относится и к аналогичной фигуре для идентификации ТС с несанкционированным внешним считывателем бесконтактного доступа.

5.4.3 Общая концепция коммуникации для удаленного доступа

Настоящий стандарт также поддерживает удаленный доступ к ERT. Регистрирующий орган может, например, использовать удаленный доступ, если он будет реализован, для проверки или обновления данных о дополнительных ТС или данных безопасности. Владелец ERT может применять удаленный доступ для проверки данных ERI или авторизации дополнительного поставщика услуг.

На рисунке 7 представлена концепция взаимосвязи для удаленного доступа к ERT ТС.

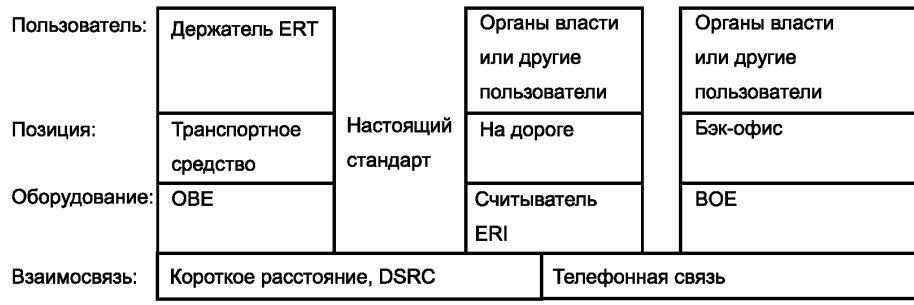

Рисунок 7 — Общая концепция дистанционной взаимосвязи

В настоящем стандарте представлен уровень приложения сетевого интерфейса между бортовым оборудованием ERI и дистанционным внешним устройством чтения/записи ERI.

Примечание — Реализация возможностей удаленного доступа выходит за рамки настоящего стандарта.

5.4.3.1 Бортовая связь

На рисунке 8 представлен абстрактный обзор возможной архитектуры бортовой взаимосвязи.

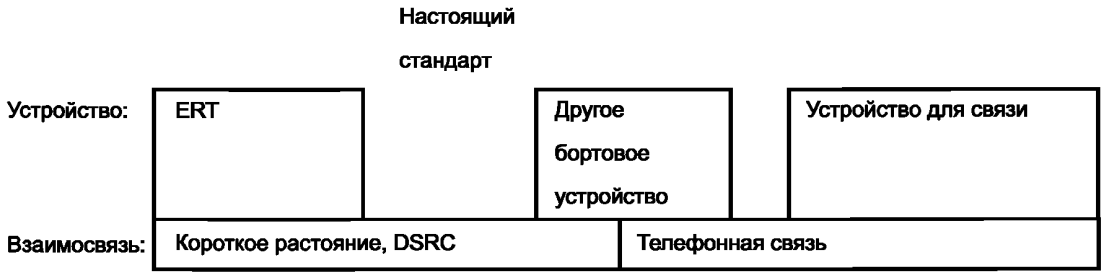

Рисунок 8 — Встроенная архитектура

Примечание — При конкретной реализации определяется, должны ли ERT и устройство связи быть отдельными компонентами.

5.5 Интерфейсы

5.5.1 Ближний беспроводной интерфейс

Взаимосвязь между бортовым оборудованием ERI и внешним считывателем/записывающим устройством ERI на коротких расстояниях использует стек протоколов, приведенный на рисунке 9.

AVI (см. [6] плюс дополнительные услуги)
Уровень ERI (добавление служб безопасности ERI и управления ими)
Уровень приложения, например прикладной уровень DSRC или аналогичный уровень
Нижние слои

Рисунок 9 — Стек протокола для ближнего беспроводного интерфейса

Примечание — Отношение между этими уровнями и опорными точками BETA к ZETA приведено на рисунке 10 (см. приложение А [5]). (Ориентир ALPHA расположен между считывающим устройством ERI и BOE бэк-офиса.)

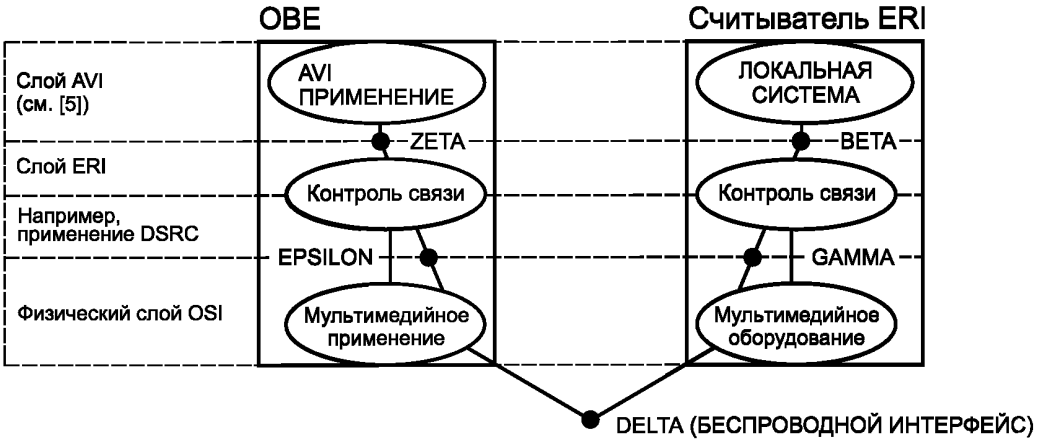

Рисунок 10 — Расположение уровня ERI в эталонной архитектуре (см. [5])

5.5.2 Бесконтактный интерфейс с ERT

Взаимосвязь между ERT и бортовым или внешним считывателем/записью проксимити использует стек протоколов, приведенный на рисунке 11.

AVI (см. [6] плюс дополнительные услуги)
Уровень ERI (добавление служб безопасности ERI и управления ими)
Уровень передачи
Нижние слои

Рисунок 11 — Стек протокола для интерфейса между ERT и считывающим устройством

6 Требования к интерфейсу

6.1 Общие положения

В настоящем разделе представлен интерфейс прикладного уровня для доступа к данным ERI в ERT, в частности:

в 6.2 приведено абстрактное определение транзакций ERI с использованием ASN.1;

6.3.2 определен бесконтактный интерфейс с ERT;

6.3.3 определен ближайший беспроводной интерфейс между бортовым оборудованием ERI и внешним считывающим устройством/считывателем ERI;

6.3.4 приведен интерфейс для удаленного доступа.

Уровень приложения встроенного интерфейса определяется как реализация абстрактных определений в 6.2. Внешние интерфейсы, приведенные в 6.3.3 и 6.3.4, используются либо для прямой связи с ERT в том случае, когда встроенное оборудование ERI интегрировано в одно устройство (ERT), либо для передачи блоков данных протокола ERI на встроенный считыватель/запись ERI (см. рисунок 1).

6.2 Абстрактные определения транзакций

6.2.1 Описание транзакций

В настоящем пункте определены транзакции ERI в таблице 3.

Таблица 3 — Обязательные и необязательные транзакции

Пункт	Транзакция	Req[a)]	Описание
Операции чтения данных ERI			
6.2.3	getEriData	R	Для ERI со стороны органов власти и дополнительных поставщиков услуг
6.2.4	getAuthenticatedEriData	C	Для получения аутентифицированных данных ERI держателем ERT
Транзакции управления данными (включая настройку) ERI			
6.2.5	setEriData	R	Для настройки ERT производителем или регистрирующим органом и для обновления дополнительных данных ТС регистрирующим органом
6.2.6	getCiphertextHistoric EriData	O	Для получения авторитетом аргумента более ранних транзакций setEriData в зашифрованном тексте
6.2.7	getCleartextHistoric EriData	O	Для получения держателем ERT аргумента предыдущих транзакций setEriData в открытом тексте
Комиссионные сделки			
6.2.8	getPublicCertificate VerificationKeyId	A, C	Для получения идентификатора открытого ключа проверки центра сертификации высшего уровня
6.2.9	getPublicEncipherment KeyErt	A, C	Для получения общедоступного ключа шифрования ERT для шифрования частного компонента данных ввода в эксплуатацию
6.2.10	commissionErt	A, C	С целью наделения регистрирующего органа полномочиями регистрации для ERT и обновления данных безопасности регистрирующего органа

Окончание таблицы 3

Пункт	Транзакция	Req[a)]	Описание
6.2.11	withdrawCommissioning	O	С целью отзыва регистрирующим органом разрешения на ввод в эксплуатацию ТС (поддержка «до конца срока службы»)
6.2.12	getCiphertextHistoric ComData	O	Для получения органом власти аргумента предыдущих транзакций CommissioningErt в зашифрованном тексте
6.2.13	getCleartextHistoric ComData	O	Для получения держателем ERT аргумента предыдущих транзакций CommissioningErt в cleartext
Операции контроля доступа			
6.2.14	updateAccessControlList	O, C	Для добавления или удаления полномочий или дополнительного поставщика услуг в/или из списка управления доступом ERT
6.2.15	getCiphertextAccess ControlListEntry	O	Для получения авторитетом данных из списка управления доступом в зашифрованном тексте
6.2.16	getCleartextAccess ControlListEntry	O	Для получения держателем ERT записи списка управления доступом в открытом тексте
Прочее			
6.2.17	getErtCapabilities	O	Для получения возможности (шифрования) ERT

[a)] Столбец, озаглавленный «Req», указывает, всегда ли требуется транзакция (R), требуемая для обеспечения конфиденциальности (C), которая предназначена для аутентификации (A), или она является необязательной (O).

Определение транзакций основано на классе информационных объектов ASN.1 TRANSACTION (см. 6.2.2).

В 6.2.18 приведены определения, являющиеся общими для нескольких транзакций.

6.2.2 PDU ERI и транзакции

6.2.2.1 Концепция транзакции

Запись и чтение данных в/из ERT достигается посредством транзакций, которые определены как экземпляры класса информационных объектов TRANSACTION.

Класс информационных объектов TRANSACTION определен следующим образом:

```
TRANSACTION ::= CLASS {
&ArgumentType                   ,
&ResultType                     ,
&ErrorCodes                     ErrorCode OPTIONAL,
&transactionCode                INTEGER UNIQUE
}
WITH SYNTAX {
ARGUMENT                        &ArgumentType
RESULT                          &ResultType
[ERRORS                         &ErrorCodes]
CODE                            &transactionCode
}
```

Сделка должна быть вызвана считывателем ERI или устройством ERI и выполнена ERT.

Сделка считается атомной в том смысле, что данные в ERT не будут изменены, если транзакция не будет выполнена (подробнее см. спецификацию отдельных транзакций).

Одна транзакция не должна вызываться встроенным устройством чтения/записи ERT в то время, когда ERT выполняет другую транзакцию. Если ответ может быть непредсказуемым, то атомарность транзакций должна быть гарантирована.

Поле &ArgumentType должно указывать тип данных аргумента транзакции.

Если транзакция выполнена успешно, она возвращает результат, указанный в поле &ResultType. В противном случае возвращается код ошибки, указанный в поле &ErrorCode.

Поле &ResultType должно указывать тип данных возвращаемого значения результата транзакции при успешной транзакции.

Поле &ErrorCode, если оно присутствует, имеет тип ErrorCode.

Поле &transactionCode указывает целочисленное значение, которое используется для идентификации типа транзакции.

6.2.2.2 Единицы данных протокола ERI

Блоки данных протокола ERI (PDU) определены следующим образом:

```
        EriPdu ::= CHOICE    {
                requestPdu     EriRequestPdu,
                reponsePdu     EriResponsePdu
                }

        EriRequestPdu ::= SEQUENCE {
                transactCode            TRANSACTION.&transactionCode                                    (
{EriTransactions}),
                argument      TRANSACTION.&ArgumentType
                        ( {EriTransactions} {@.transactCode}) OPTIONAL
                }

        EriResponsePdu ::= CHOICE {
                resultPdu      EriResultPdu,
                errorPdu       EriErrorPdu
                }

        EriResultPdu ::= SEQUENCE {
                transactCode            TRANSACTION.&transactionCode                                    (
{EriTransactions}),
                result          TRANSACTION.&ResultType                 (                       {EriTransactions}
{@.transactCode})
                }

        EriErrorPdu ::= SEQUENCE {
                transactCode            TRANSACTION.&transactionCode                                    (
{EriTransactions}),
                error           TRANSACTION.&ErrorCodes                 (                       {EriTransactions}
{@.transactCode})
                }

        EriTransactions
TRANSACTION             ::=         {
        getEriData                  |
getAuthenticatedEriData    |
                setEriData                  |               getCiphertextHistoricEriData                 |
getCleartextHistoricEriData             |               getPublicCertificateVerificationKeyId
| getPublicEnciphermentKeyErt    |
```

```
            commissionErt | withdrawCommissioning |
            getCiphertextHistoricComData | getCleartextHistoricComData                    |
            updateAccessControlList                                                        |
getCiphertextAccessControlListEntry | getCleartextAccessControlListEntry
|       getErtCapabilities
            }
```

Блок данных протокола ERI имеет тип EriData и является либо EriRequestPdu, либо EriResponse Pdu.

Блок данных протокола EriRequestPdu используется для вызова транзакции, которая должна быть выполнена ERT.

Единица протокола данных EriResponsePdu используется для результата или уведомления об ошибке транзакции, выполняемой ERT.

EriTransactions задает набор транзакций ERI.

Компонент transactCode должен содержать значение кода транзакции для транзакции в установленных EriTransactions.

Компонент аргумента, при его наличии, должен быть того же типа, который указан для аргумента транзакции, идентифицируемой значением transactCode.

Компонент аргумента должен присутствовать, если аргумент для транзакции определен, и отсутствовать, если не определен.

Компонент результата должен быть того же типа, что и для результата транзакции, идентифицированного значением transactCode.

Компонент ошибки должен быть кодом ошибки из набора кодов ошибок, как указано для транзакции, идентифицированной значением transactCode.

6.2.3 Получить данные ERI

6.2.3.1 Операция получения данных ERI

Операция getEriData используется для чтения данных ERI с помощью считывателя ERI. Операция getEriData определена следующим образом:

```
        getEriData TRANSACTION
::= {
            ARGUMENT            GetEriDataArgument
            RESULT              GetEriDataResult
            ERRORS              {notCustomized}
            CODE                1
            }
```

Операция getEriData вызывается с аргументом GetEriDataArgument и возвращает результат GetEriDataResult.

Когда ERT еще не настроена, возвращается код ошибки notCustomized.

Операция getEriData является единственной критически важной транзакцией ERI.

Примечание — Это единственная транзакция, которая должна быть выполнена во время движения ТС. Кроме того, отсутствует возможность предотвратить идентификацию посредством высокой скорости ТС.

6.2.3.2 Аргумент транзакции данных ERI

Тип GetEriDataArgument определен следующим образом:

```
        GetEriDataArgument ::=
SEQUENCE {
            onBehalfOf              EntityId OPTIONAL,
            challenge               Challenge,
        includeAdditionalData       BOOLEAN
            }
```

Компонент onBehalfOf, при его наличии, указывает объект, открытый ключ шифрования которого в конечном итоге будет использован для возвращаемых данных. В противном случае, когда ERT вводится в эксплуатацию, значением по умолчанию является EntityId регистрирующего органа. Когда ERT еще не введена в эксплуатацию, значение компонента onBehalfOf игнорируется.

Примечание — Компонент onBehalfOf идентифицирует либо регистрирующий орган, либо дополнительный поставщик услуг.

Компонент вызова — это случайное число, генерируемое считывателем ERI, которое в результате должно быть возвращено для того, чтобы показать, что возвращенный результат действительно является результатом этого вызова, а не реплицированного сообщения.

Компонент includeAdditionalData должен быть использован для того, чтобы указать, будет ли результат содержать дополнительные данные ТС. Если использован FALSE, то включен только идентификатор ТС; если TRUE, то также должны быть включены дополнительные данные ТС (при их наличии), предоставленные регистрирующим органом ТС.

6.2.3.3 Результат транзакции данных ERI

Тип GetEriDataResult определен следующим образом:

```
GetEriDataResult ::= SEQUENCE {
registrationAuthority                EntityId OPTIONAL,
    eriResultData                    SECURED {CleartextEriData}}
    }
```

Компонент registrationAuthority, при его наличии, должен идентифицировать регистрирующий орган ТС.

Компонент registrationAuthority должен присутствовать, когда ERT вводится в эксплуатацию, и отсутствовать в обратном случае.

Компонент eriResultData должен содержать защищенные данные ERI открытого текста (см. 6.2.18.1).

Четкие данные ERI должны быть защищены, т. е. подписаны и/или зашифрованы, в соответствии с возможностями ERT.

6.2.3.4 Данные Cleartext ERI

Тип CleartextEriData используется для данных ERI в соответствии с запросом на вызов транзакции getEriData или GetAuthenticatedEriData и определяется следующим образом:

```
    CleartextEriData        ::=
SEQUENCE {
        eriDataOrId                  EriDataOrId,
        ErtSecurityStatus            ErtSecurityFlags OPTIONAL
        }
```

Компонент eriDataOrId имеет тип EriDataOrId, как определено ниже.

Компонент ertSecurityStatus, при его наличии, должен иметь тип ErtSecurityFlags, как определено ниже.

Компонент ertSecurityStatus должен присутствовать, если ERT имеет возможность установить или сбросить как минимум один из битов состояния безопасности. В противном случае компонент не должен присутствовать.

6.2.3.5 Данные или идентификатор ERI

Тип EriDataOrId используется для указания данных VehicleId или ERI и определяется следующим образом:

```
EriDataOrId ::= CHOICE {
    vehicleId                        VehicleId,
unsignedDatedEriData                 DATED {EriData},
datedAndSignedEriData                SIGNED {DATED {EriData}, PrivateSignatureKey} -- BOE signed
    }
```

Альтернатива VehicleId должна использоваться, если компонент includeAdditionalData в поле аргумента вызываемой транзакции имеет значение FALSE.

Компонент unsignedDatedEriData должен использоваться, если компонент includeAdditionalData в поле аргумента вызываемой транзакции имеет значение TRUE и данные ERI не подписаны при записи в ERT.

Значение компонента unsignedDatedEriData является значением данных, полученных от ERI, как записано в ERT при выполнении последней транзакции данных ERI.

Компонент dateAndSignedEriData должен использоваться, если компонент includeAdditionalData в поле аргумента вызываемой транзакции имеет значение TRUE и данные ERI подписаны при записи в ERT.

Значение компонента dateAndSignedEriData должно быть значением устаревших и подписанных данных ERI, записанных в ERT последней успешно выполненной транзакцией данных ERI.

6.2.3.6 Данные ERI

Тип EriData должен использоваться для данных ERI и определяется следующим образом:

```
EriData ::= SEQUENCE {
    id                      VehicleId,
    additionalEriData       OCTET          STRING          (CONTAINING
    }                   AdditionalEriData) OPTIONAL
```

Идентификатор должен содержать идентификатор ТС в соответствии с ПНСТ 343—2018. Дополнительным компонентом EriData является строка октета, содержащая дополнительные данные ERI согласно ПНСТ 343—2018.

Примечание — При переносе дополнительных данных ERI в строку октета ERT не нуждается в кодировании/декодировании дополнительных данных ERI. Поэтому будущие изменения определения дополнительного типа данных ERI также не будут влиять на дизайн ERT (кроме максимального размера записи данных ERI).

6.2.3.7 Флаги безопасности ERT

Целью флагов безопасности ERT является сигнализация безопасности, которая может быть вызвана попытками вмешаться в ERT.

Примечание — Возможности обеспечения безопасности ERT можно получить с помощью транзакций получения возможностей ERT. Тип ErtSecurityFlags используется для указания флагов безопасности и определяется следующим образом:

```
ErtSecurityFlags ::= BIT STRING {
        flagsHaveBeenResetted     (0),
        notCommissioned           (1),
        lowSupplyVoltageIndication (2),
        highSupplyVoltageIndication
                                  (3),
        lowClockFrequencyIndication
                                  (4),
        highClockFrequencyIndication
                                  (5),
        lowTemperatureIndication  (6),
        highTemperatureIndication (7)
        } (SIZE (0..16))    -- bit 8 .. 15 reserved for future use
```

Начальное значение битовой строки должно быть «0100000000000000» B, т. е. еще не введено в эксплуатацию, еще не обнаружено и сброс отсутствует.

Если значение типа ErtSecurityFlags равно ‘0000000000000000’B или ‘0100000000000000’B, операция сброса не изменит это значение. В противном случае операция сброса установит его на «1000000000000000».

Бит notCommissioned будет установлен в 1, если транзакция ввода в эксплуатацию будет выполнена успешно, и установлен в 0, если вызывается транзакция ввода/вывода.

Примечание — Значение компонента ErtSecurityFlags можно сбросить с помощью транзакции ERT комиссии.

Бит 2 .. 15 будет установлен в 1, если соответствующее условие обнаружено.

Примечание — Значение 0 указывает на то, что условие не обнаружено. Это может быть связано с тем, что ERT не имеет соответствующей возможности обнаружения (см. также 6.2.17).

6.2.4 Аутентифицированные данные ERI

6.2.4.1 Аутентифицированная транзакция данных ERI

Операция getAuthenticatedEriData используется для получения аутентифицированной версии данных ERI и определяется следующим образом:

```
getAuthenticatedEriData  TRANSACTION ::= {
    ARGUMENT                GetAuthenticatedEriDataAr
                        gument
    RESULT                  GetAuthenticatedEriDataRe
                        sult
    ERRORS                  {notCustomized}
    CODE                    2
    }
```

Операция getAuthenticatedEriData вызывается с аргументом GetAuthenticatedEriDataArgument.
Если ERT настроена, возвращается GetAuthenticatedEriDataResult.
Если ERT не настроена, возвращается код нестандартной ошибки.

6.2.4.2 Аутентифицированный аргумент транзакции данных ERI

Тип GetAuthenticatedEriDataArgument определен следующим образом.

```
GetAuthenticatedEriDataArgument ::= SEQUENCE {
    ertHolderCredentials    ErtHolderCredentials,
    challenge   Challenge,
includeAdditionalData       BOOLEAN
    }
```

Компонент ertHolderCredentials должен содержать учетные данные владельца ERT, т. е. идентификатор ТС и PIN-код владельца ERT.

Компонент вызова и компонент includeAdditionalData определены так же, как и в типе GetEriDataArgument.

6.2.4.3 Результат аутентификации данных ERI

Тип GetAuthenticatedEriDataResult определен следующим образом:

```
GetAuthenticatedEriDataResult ::=  SEQUENCE {
    registrationAutho                   EntityId OPTIONAL,
rity
    authenticateResul                   CLEAR-SECURED
tData                           {CleartextEriData}
    }
```

Компонент registrationAuthority, при его наличии, должен идентифицировать регистрирующий орган ТС.

Компонент registrationAuthority должен присутствовать, если ERT введен в эксплуатацию, и отсутствовать в противном случае.

Элемент authenticateResultData должен содержать защищенные, но не зашифрованные данные ERI открытого текста (см. 6.2.18.1).

Четкие данные ERI четко защищены, т. е. подписаны/не подписаны, в соответствии с возможностями ERT.

6.2.5 Установка данных ERI

6.2.5.1 Установленная транзакция данных ERI

Операция setEriData используется для настройки и обновления ERT и определяется следующим образом:

```
setEriData TRANSACTION ::= {
        ARGUMENT                    SetEriDataArgument
        RESULT                      NULL
        ERRORS                      {SetEriDataErrors}
        CODE                        3
        }
```

setEriData вызывается с аргументом типа SetEriDataArgument.

Если транзакция выполнена успешно, ERT возвращает значение NULL. В противном случае возвращается один из кодов ошибок из SetEriDataErrors.

Данные не должны храниться в ERT, если транзакция не выполнена успешно.

Когда транзакция выполняется в первый раз с помощью ERT, считается, что ERT настроена. Когда ERT настроена, TC в данной ERT не может быть изменено ни при каких обстоятельствах.

Примечание — Если ERT не имеет возможности проверки подписи, любой писатель может записывать любые данные ERI в ERT.

6.2.5.2 Аргумент транзакции данных набора ERI

Тип SetEriDataArgument определен следующим образом:

```
SetEriDataArgument ::= CHOICE {
        clearTextArgument           ClearTextSetEriDataArgument,
        encryptedArgument           ENCRYPTED {ClearTextSetEriDataArgument}
        }
```

Альтернатива зашифрованного аргумента не выбирается, если у ERT отсутствуют возможности дешифрования.

Примечание — Независимо от того, может ли ERT определить возможности дешифрования с помощью транзакции получения ERT.

Если выбрана альтернатива encryptment, значение ClearTextSetEriDataArgument должно быть зашифровано с помощью ключа общедоступного шифрования, который может быть получен с транзакцией getPublicEnciphermentKeyErt.

6.2.5.3 Тип аргумента данных типа ERI

Тип ClearTextSetEriDataArgument определен следующим образом:

```
ClearTextSetEriDataArgument ::= CHOICE {
        authenticatedEriData        BOE-AUTHENTICATED {DATED {EriData}},
notAuthenticatedEriData    DATED {EriData}
        }
```

Аутентифицированная альтернатива EriData выбирается автором ERI в тот момент, когда ERT имеет возможности проверки подписи, и может быть выбрана, когда ERT не имеет возможностей проверки подписи. NotAuthenticatedEriData может быть выбрана только в том случае, если ERT не имеет возможностей проверки подписи.

Примечание — Независимо от того, может ли ERT иметь возможности проверки подписи, можно определить транзакцию get ERT.

Аутентифицированные EriData, при их наличии, должны содержать данные ERI как датированные и подписанные [см. перечисление а) 6.2.18.3] регистрирующим органом или изготовителем, включая сертификат(ы) для открытого ключа проверки регистрирующего органа или производителя.

NotAuthenticatedEriData, при его наличии, должен содержать данные ERI, устаревшие [см. перечисление а) 6.2.18.3].

Когда транзакция вызывается в течение второго или последующего времени, TC в аргументе должно быть таким же, как и при настройке ERT.

6.2.5.4 Регистрация заданных аргументов данных ERI

Значение ClearTextSetEriDataArgument каждой транзакции setEriData должно быть последовательно пронумеровано и сохранено в ERT для последующего извлечения. Значение ClearText SetEriDataArgument транзакции настройки должно быть пронумеровано 1.

ERT должна иметь возможность сохранять одно или несколько значений ClearTextSetEriDataArgument, т. е. ноль значений или более от успешно выполненных ранее установленных вызовов данных ERI.

Если достигнуто максимальное количество значений ClearTextSetEriDataArgument, которые удалось сохранить ERT, применяется следующая процедура:

когда ERT может хранить только одно значение ClearTextSetEriDataArgument, т. е. исторические данные не хранятся, новое значение ClearTextSetEriDataArgument заменяет старое;

когда ERT может хранить два значения ClearTextSetEriDataArgument или более, т. е. хранить исторические данные, новое значение ClearTextSetEriDataArgument заменяет второе старое существующее значение ClearTextSetEriDataArgument.

Примечание — Удаляя второе старое, а не самое старое значение, исходные данные ERI сохраняются.

Если размер значения ClearTextSetEriDataArgument превышает максимальное значение, разрешенное для этого аргумента, возвращается код ошибки resourceLimitExceeded и обновление данных ERI не происходит.

6.2.5.5 Аутентификация данных ERI

Данные аутентификации не должны иметь компонент даты с датой, предшествующей дате предыдущего вызова setEriData.

Примечание — Это также защита от повторной атаки, предотвращающая незаконную переустановку исторической версии данных ERI несанкционированной записи ERI.

Если транзакция вызывается на второе время или последующее до тех пор, пока ERT не введена в эксплуатацию, значение компонента эмитента в датированных данных должно быть таким же, как и в транзакции настройки (т. е. от того же производителя или регистрирующего органа).

Если транзакция вызывается при вводе ERT, объект EntityId в данных аутентификации должен быть таким же, как и в транзакции ввода в эксплуатацию (т. е. от того же регистрирующего органа).

Сделка не должна вызываться во время снятия ввода в эксплуатацию.

Когда у ERT есть возможности проверки подписи, применяются следующие условия:

- если ERT не вводилась в эксплуатацию, данные в аргументе подписываются изготовителем или регистрирующим органом;
- если ERT была введена в эксплуатацию как минимум один раз, данные в аргументе должны быть подписаны регистрирующим органом;
- доступ к ERT контролируется сертификатами, включенными в аргумент. Все сертификаты в аргументе должны быть действительными (т. е. уже выпущены и еще не истекли) на дату, содержащуюся в компоненте даты в данных аутентификации.

6.2.5.6 Установленные ошибки транзакции данных ERI

Тип SetEriDataErrors является подтипом типа ErrorCode, который определен следующим образом:

```
SetEriDataErrors ErrorCode ::= {
        illegalArgument |                illegalVehicleId |
        illegalCertificate |             illegalSignature |
        illegalDate |                    notCommissioned |
resourceLimitExceeded |                  otherError
        }
```

SetEriDataErrors имеют тип EriErrorCode и содержат перечисленные значения.

Когда ERT имеет возможности проверки подписи и данные ERI в аргументе не аутентифицированы, возвращается значение незаконного аргумента.

Когда транзакция вызывается во время снятия ввода в эксплуатацию, возвращается значение notCommissioned.

6.2.6 Получать исторические данные ERI зашифрованного текста

6.2.6.1 Передача данных об аутентификации ERI зашифрованного текста

Операция getCiphertextHistoricEriData используется для извлечения окончательно зашифрованного аргумента предыдущей транзакции setEriData и определяется следующим образом:

```
getCiphertextHistoricEriData TRANSACTION ::= {
  ARGUMENT      GetCiphertextHistoricEriDataArgument
  RESULT                  SECURED {HistoricEriData}
  ERRORS        {notCustomized}
  CODE                    4
      }
```

Операция getCiphertextHistoricEriData вызывается с аргументом GetCiphertextHistoricEriDataArgument.

Если ERT настроена, транзакция возвращает исторические данные ERI, подписанные и/или зашифрованные в соответствии с возможностями ERT (см. 6.2.18.1 для определения SECURED {}).

Если ERT еще не настроена, возвращается код с нестандартной ошибкой.

6.2.6.2 Аргумент транзакции данных ERI получения зашифрованного текста

Тип GetCiphertextHistoricEriDataArgument определен следующим образом:

```
GetCiphertextHistoricEriDataArgument ::= SEQUENCE {
    onBehalfOf              EntityId OPTIONAL,
    challenge               Challenge,
    number                  INTEGER (1..int4)
        }
```

Компонент onBehalfOf, при его наличии, указывает объект, открытый ключ шифрования которого в конечном итоге будет использован для возвращаемых данных. Если это не указано, значением по умолчанию является объект EntityId регистрирующего органа ТС. Когда ERT еще не введена в эксплуатацию, значение компонента onBehalfOf игнорируется.

Компонент вызова — это случайное число, генерируемое считывателем ERI, которое в конечном итоге должно быть возвращено в подписанной части результата, для того чтобы показать, что возвращенный результат является результатом этого вызова, а не реплицированного сообщения.

Сохраненные аргументы транзакций setEriData последовательно нумеруются. Первый сохраненный аргумент (для настройки транзакции setEriData) пронумерован 1.

Компонент number идентифицирует аргумент вызова setEriData.

Когда значением числового компонента является число удаленных аргументов, возвращается самый старый доступный аргумент перед удаленной транзакцией setEriData.

Когда значение числового компонента более числа последнего сохраненного аргумента, данные этого последнего аргумента должны быть возвращены.

6.2.6.3 Исторические данные ERI

Тип HistoricEriData используется для извлечения аргументов и для установки транзакций данных ERI и определяется следующим образом:

```
    HistoricEriData     ::=
SEQUENCE {
        number                  INTEGER (1..int4),
        outOf                   INTEGER (1..int4),
        historicRecord          ClearTextSetEriDataArgument
        }
```

Компонент числа HistoricEriData идентифицирует значение ClearTextSetEriDataArgument, переданное в компоненте HistoricalRecord.

Компонент outOf указывает число самого последнего сохраненного значения ClearText SetEriDataArgument.

Компонент historyRecord содержит значение ClearTextSetEriDataArgument, определенное значением числового компонента.

6.2.7 Получение исторических данных ERI с открытым текстом

6.2.7.1 Исходная транзакция данных ERI

Операция getCleartextHistoricEriData должна использоваться для извлечения аргумента предыдущих транзакций setEriData в открытом тексте и определяется следующим образом:

```
getCleartextHistoricEriData TRANSACTION  ::=  {
        ARGUMENT                GetCleartextHistoricEriDataArgument
        RESULT                  CLEAR-SECURED {HistoricEriData}
        ERRORS                  {notCustomized}
        CODE                    5
        }
```

Поле ARGUMENT имеет тип GetCleartextHistoricEriDataArgument.

Транзакция возвращает исторические данные ERI (см. 6.2.6.3), подписанные/неподписанные в соответствии с возможностями ERT (см. 6.2.18.1 при определении CLEAR-SECURED {}).

Если ERT еще не настроена, возвращается код с нестандартной ошибкой.

6.2.7.2 Исходный аргумент транзакции данных ERI

Тип GetCleartextHistoricEriDataArgument определен следующим образом:

```
GetCleartextHistoricEriDataArgument  ::=  SEQUENCE {
        credentials             ErtHolderCredentials,
        challenge               Challenge,
        number                  INTEGER (1..int4)
        }
```

Компонент учетных данных определяется так же, как и в аргументе транзакции getAuthenticatedEriData.

Компоненты вызова и числа определяются так же, как и в аргументе транзакции getCiphertextHistoricEriData.

6.2.8 Получение транзакции с идентификатором ключа проверки подлинности сертификата

Операция getPublicCertificateVerificationKeyId используется для извлечения идентификатора открытого ключа проверки, который будет использоваться для проверки сертификатов открытого ключа высшего уровня. Операция getPublicCertificateVerificationKeyId определяется следующим образом:

```
getPublicCertificateVerificationKeyId TRANSACTION ::= {
ARGUMENT NULL RESULT KeyId
        CODE   6
        }
```

Поле аргумента имеет тип NULL.

Результатом транзакции является идентификатор открытого ключа проверки, который будет использоваться для проверки сертификатов открытого ключа toplevel. Если ERT не поддерживает проверку подписи или этот ключ недоступен, возвращаемое значение равно нулю.

Примечание — Это позволяет центру сертификации высшего уровня использовать несколько пар ключей для своих сертификатов. Затем ключевой идентификатор можно использовать для выбора достоверного(ых) сертификата(ов) для транзакции записи.

6.2.9 Получение общедоступного ключа шифрования ERT

6.2.9.1 Получение транзакции общедоступного ключа шифрования ERT

Транзакция getPublicEnciphermentKeyErt должна использоваться для извлечения ключа общедоступного шифрования ERT регистрирующим органом. Этот ключ необходим для шифрования данных личного ввода в эксплуатацию. Операция getPublicEnciphermentKeyErt определена следующим образом:

```
getPublicEnciphermentKeyErt TRANSACTION ::= {
    ARGUMENT            BOE-AUTHENTICATED
                        {vehicleId}
    RESULT              PublicEnciphermentKey
    ERRORS              {GetPublicEnciphermentKeyE
                        rrors}
    CODE                6
    }
```

Поле аргумента имеет тип BOE-AUTHENTICATED {VehicleId} и содержит идентификатор VehicleId как аутентифицированное BOE (см. 6.2.18.4). Подписчиком данных аутентификации является регистрирующий орган.

Примечание — Так как ключ общедоступного шифрования ERT используется для шифрования других ключей, это — главный ключ, который должен быть недоступен для объектов без реальной необходимости.

В случае успеха транзакция возвращает общедоступный ключ шифрования для ERT. Если это не удается, транзакция возвращает значение GetPublicEnciphermentKeyErrors.

6.2.9.2 Получение ошибок публичного шифрования транзакций ERT

Тип GetPublicEnciphermentKeyErrors — это подтип типа ErrorCode, который определяется следующим образом:

```
    GetPublicEnciphermentKeyErrors ErrorCode ::= {
        illegalArgument             |
    illegalVehicleId                |
    illegalCertificate              |
    illegalSignature    |   illegalEntity
|   noDeciphermentCapability    |
        otherError }
```

Тип GetPublicEnciphermentKeyErrors имеет тип EriErrorCode и должен принимать одно из перечисленных значений.

6.2.10 Комиссия ERT

6.2.10.1 Комиссия по сделке ERT

Регистрирующий орган должен использовать транзакцию commErt для поручения ERT или обновления параметров безопасности ERT. Сделка commissionErt определена следующим образом:

```
commissionErt TRANSACTION ::=
{
    ARGUMENT                    CommissionErtArgument
    RESULT                      NULL
    ERRORS                      { CommissionErtErrors }
    CODE                        7
    }
```

Примечание — Сделка ввода в эксплуатацию может быть вызвана только при настройке ERT.

Аргумент comissionErt транзакции имеет тип CommissionErtArgument.

Если транзакция выполнена успешно, ERT возвращает значение NULL. В противном случае возвращается один из кодов ошибок CommissionErtErrors.

Данные не должны храниться в ERT, если транзакция не выполнена успешно.

Примечание — Если ERT не имеет возможности проверки подписи, любой автор может записать в ERT данные ввода в эксплуатацию.

6.2.10.2 Аргумент ввода в эксплуатацию ERT

Тип CommissionErtArgument определен следующим образом:

```
CommissionErtArgument ::= CHOICE {
        authenticatedData                BOE-AUTHENTICATED { DATED {CommissioningData}},
        notAuthenticatedData             DATED {CommissioningData}
        }
```

Альтернатива аутентифицированной альтернативы выбирается, когда ERT имеет возможность проверки подписи, и может быть выбрана, когда ERT не имеет возможностей проверки подписи. NotAuthenticatedData может быть выбрана только в том случае, если ERT не имеет возможности проверки подписи.

Аутентифицированные данные, при их наличии, должны содержать данные о вводе в эксплуатацию, как датированные и подписанные [см. перечисление а) 6.2.18.3 и 6.2.18.3] регистрирующим органом, включая сертификат(ы) для открытого ключа проверки регистрирующего органа.

Неаутентифицированные данные, при их наличии, должны содержать данные о вводе в эксплуатацию [см. перечисление а) 6.2.18.3] регистрирующим органом.

6.2.10.3 Регистрация ввода в эксплуатацию ERT и снятия аргументов ввода в эксплуатацию

Аргументы комиссии ERT и снятия транзакций ввода в эксплуатацию, которые выполняются успешно, должны быть последовательно пронумерованы и сохранены для последующего поиска. Аргумент первого успешного вызова должен быть пронумерован 1.

ERT должна быть способна хранить одну или несколько комиссионных ERT и снимать аргументы о вводе в эксплуатацию, т. е. ноль аргументов или более успешно выполненной предыдущей комиссии ERT, или аннулировать транзакции.

Если достигнуто максимальное количество транзакций ERT и снятие комиссионных операций, которые удалось сохранить ERT, применяется следующая процедура:

если ERT может хранить только один аргумент, т. е. исторические данные не хранятся, новый аргумент заменяет старый;

если ERT может хранить два аргумента или более, т. е. хранить исторические данные, новый аргумент заменяет второй самый старый аргумент.

Если размер аргумента превышает максимальный размер, который может хранить ERT, возвращается код ошибки resourceLimitExceeded.

6.2.10.4 Данные для ввода в эксплуатацию

Тип ввода в эксплуатацию содержит все данные, необходимые для ввода в эксплуатацию ERT, и определяется следующим образом:

```
CommissioningData ::= SEQUENCE {
        vehicleId           VehicleId,
registrationAuthority EntityId, resetSecurityFlags BOOLEAN,
        enciphermentKeyId           KeyId OPTIONAL,
        publicEnciphermentKeyAuthority PublicEnciphermentKey OPTIONAL,
        publicVerificationKeyCertificate      ErtCertificate OPTIONAL,
        privateData         ENCRYPTED {PrivateCommissioningData} OPTIONAL
        }
```

Компонент vehicleId идентифицирует ТС. Значение должно быть таким же, как и для настройки ERT.

Компонент registrationAuthority идентифицирует регистрирующий орган, который (повторно) поручил ERT.

В компоненте resetSecurityFlags указывается, следует ли сбросить флаги безопасности: если TRUE, они должны быть сброшены; если FALSE, они не будут сброшены.

Компонент enciphermentKeyId, при его наличии, идентифицирует открытый ключ шифрования, используемый для шифрования. Различные регистрирующие органы могут использовать одни и те же идентификаторы.

Примечание — При возврате с зашифрованными данными это значение может использоваться для определения соответствующего частного ключа дешифрования.

Компонент enciphermentKeyId должен присутствовать, если присутствует компонент publicEnciphermentKeyAuthority. В противном случае компонент должен отсутствовать.

Компонент publicEnciphermentKeyAuthority, при его наличии, содержит открытый ключ шифрования регистрирующего органа, указанный компонентом registrationAuthority. Данный ключ используется для шифрования результатов, возвращаемых ERT, если это применимо.

Компонент publicEnciphermentKeyAuthority должен присутствовать, когда ERT имеет возможности шифрования, которые должны быть включены. В противном случае компонент должен отсутствовать, и любые возможности шифрования ERT, при их наличии, должны быть отключены.

Примечание — Это позволяет использовать любые возможности шифрования ERT, при их наличии, под контролем регистрирующего органа ввода в эксплуатацию.

Компонент publicVerificationKeyCertificate, при его наличии, должен содержать сертификат для общедоступного ключа проверки для проверки подписей ERT. Этот сертификат выдается регистрирующим органом, указанным в компоненте registrationAuthority.

Компонент publicVerificationKeyCertificate должен присутствовать, если компонент privateSignatureKeyErt присутствует в компоненте privateData. В противном случае он отсутствует.

Компонент privateData, при его наличии, содержит зашифрованные PrivateCommissioningData. Используемый общедоступный ключ шифрования должен быть PublicEnciphermentKey, который может быть получен с помощью транзакции getPublicEnciphermentKeyErt.

Компонент privateData должен присутствовать, если тип PrivateCommissioningData содержит как минимум один компонент. В противном случае он отсутствует.

6.2.10.5 Данные личного ввода в эксплуатацию

Тип PrivateCommissioningData содержит все личные данные, необходимые для ввода в эксплуатацию ERT, и определяется следующим образом:

```
PrivateCommissioningData ::= SEQUENCE {
        privateSignatureKeyErt     PrivateSignatureKey OPTIONAL,
        pin        PIN OPTIONAL   }
```

Компонент privateSignatureKeyErt, при его наличии, содержит ключ частной подписи, который будет использоваться ERT для подписи результата транзакций.

Компонент privateSignatureKeyErt должен присутствовать, когда ERT имеет возможности подписывания, которые должны быть включены. В противном случае компонент должен отсутствовать, и любые возможности подписки, при их наличии, должны быть отключены.

Примечание — Идентификатор ключа не требуется. ERT всегда прикрепляет сертификат для соответствующего открытого ключа проверки к каждой подписи ERT.

Контактный компонент, при его наличии, должен содержать случайный PIN-код, который может использоваться в сочетании с ТС, для того чтобы получить доступ к данным ERI с открытым текстом (например, держателем ERT).

Примечание — Безопасное распространение PIN-кода на держатель ERT выходит за рамки настоящего стандарта.

Компонент штыря должен присутствовать, если возможности шифрования ERT включены и должны поддерживаться getAuthenticatedEriData, getCleartextHistoricEriData, UpdateAccesControlList (для держателей) или getCleartextAccessControlListEntry. В противном случае он должен отсутствовать.

Если контактный компонент присутствует, когда возможности шифрования ERT отключены, его значение игнорируется.

6.2.10.6 Ошибки транзакции комиссии ERT

Тип CommissionErtErrors определен следующим образом:

```
        CommissionErtErrors ErrorCode ::= {
                illegalArgument                 |
        illegalVehicleId                        |
        illegalCertificate                      |
        illegalSignature |          illegalEntity
|       illegalDate |       notCustomized      |
        resourceLimitExceeded                   |
        noEnciphermentCapability                |
        secretKeyEncryptionAlgorithmNotSuppor
ted                                             |
        publicKeyEncryptionAlgorithmNotSuppor
ted                                             |
                noSigningCapability     |
        hashingAlgorithmNotSupported    |
        signingAlgorithmNotSupported    |
                otherError      }
```

Тип CommissionErtErrors имеет тип ErrorCode и должен принимать одно из перечисленных значений.

6.2.11 Ввод в эксплуатацию

6.2.11.1 Операция снятия взыскания

Операция снятия с производства должна быть использована для снятия ввода в эксплуатацию ERT, например, в случае окончания срока службы ТС.

Операция снятия с производства определена следующим образом:

```
withdrawCommissioning TRANSACTION ::= {
        -- this transaction also removes all public encipherment keys
    ARGUMENT            WithdrawCommissioningArgument
    RESULT              SECURED {WithdrawCommissioningResultData}
    ERRORS              {WithdrawCommissioningErrors}
    CODE                withdrawCommissioningCode
  }
withdrawCommissioningCode INTEGER ::= 9
```

Поле ARGUMENT должно иметь тип WithdrawCommissioningArgument.

Если ERT введена в эксплуатацию, транзакция возвращает SECURED WithdrawCommissioning-ResultData (см. ниже), подписанный и/или зашифрованный в соответствии с возможностями ERT (см. 6.2.18.1 для определения SECURED {}). В противном случае возвращается один из кодов ошибок из CommissionErtErrors.

Регистрирующий орган, который заказал ERT, может быть вызван транзакцией.

Выполнение этой транзакции приведет:

к удалению всех ключей и PIN-кода, при их наличии, установленных регистрирующим органом посредством транзакции ввода в эксплуатацию;

удалению всех записей из списка управления доступом;

отключению, в случае применения, возможностей шифрования ERT;
установке битов «notCommissioned» в флагах безопасности ERT;
нумерации и сохранению значения аргумента (см. 6.2.10.3).

Примечание — Основная идея этой транзакции — отключить дальнейшее использование ERT. Однако, чтобы предотвратить незаконное использование ТС после этой транзакции, его производительность сигнализируется только в флагах безопасности ERT, и ERT в действительности не отключена. И так как необходимость защиты конфиденциальности отсутствует, все возможности шифрования также отключены.

6.2.11.2 Аргумент транзакции снятия вводом
Тип WithdrawCommissioningArgument определен следующим образом:

```
WithdrawCommissioningArgument ::= CHOICE {
authenticatedData BOE-AUTHENTICATED {VehicleId},
NotAuthenticatedData VehicleId
}
```

Альтернатива authenticatedData выбирается в том случае, когда ERT имеет возможности проверки подписи, и может быть выбрана тогда, когда ERT не имеет данных возможностей. NotAuthenticatedData может быть выбран только при условии отсутствия у ERT возможности проверки подписи.

Аутентифицированные данные, при их наличии, должны содержать идентификатор ТС, подписанный (см. 6.2.18.4) регистрирующим органом, включая сертификат(ы) для открытого ключа проверки регистрирующего органа.

Неаутентифицированные данные, при их наличии, должны содержать идентификатор ТС.

Примечание — Если ERT не имеет возможности проверки подписи, каждый может снять ввод в эксплуатацию, даже если выбрана альтернатива authenticatedData (если подпись не подтверждена, можно использовать фиктивную подпись).

6.2.11.3 Данные результата транзакции снятия вводом
Тип WithdrawCommissioningResultData определен следующим образом:

```
WithdrawCommissioningResultData ::= [APPLICATION
withdrawCommissioningCode ] SEQUENCE {
withdrawn WithdrawCommissioningArgument,
historicComData HistoricComData
}
```

Отобранный компонент должен использоваться для возврата аргумента транзакции.

Компонент HistoricalComData должен содержать значение самого последнего хранимого аргумента транзакции ERT для ввода в эксплуатацию.

6.2.11.4 Ошибки транзакции снятия с производства
Тип WithdrawCommissioningErrors определен следующим образом:

```
WithdrawCommissioningErrors ErrorCode ::= {
illegalArgumenti |
IlegalVehicleId |
illegalCertificate |
illegalSignature | illegalEntity
| illegalDate |
notCustomized | notCommissioned |
otherError }
```

Тип WithdrawCommissioningErrors имеет тип EriErrorCode и должен принимать одно из перечисленных значений.

6.2.12 Получение данных ввода в зашифрованный текст

6.2.12.1 Получение транзакции с данными об аутентификации зашифрованного текста

Операция getCiphertextHistoricComData используется для извлечения окончательно зашифрованного аргумента предыдущей транзакции CommissionErt и определяется следующим образом:

```
getCiphertextHistoricComData TRANSACTION ::= {
    ARGUMENT            GetCiphertextHistoricComDataAr
                        gument
    RESULT              SECURED {HistoricComData}
    ERRORS              {notCommissioned}
    CODE                12
  }
```

Поле ARGUMENT должно быть типа GetCiphertextHistoricComDataArgument.

При наличии ERT или ее вводе в эксплуатацию транзакция возвращает исторические данные ввода в эксплуатацию, подписанные и/или зашифрованные в соответствии с возможностями ERT (см. 6.2.18.1 для определения SECURED {}).

Если ERT не введена в эксплуатацию, должен быть возвращен код ошибки notCommissioned.

6.2.12.2 Аргумент транзакции данных ввода данных зашифрованного текста

Тип GetCiphertextHistoricEriDataArgument определен следующим образом:

```
GetCiphertextHistoricComDataArgument ::= SEQUENCE {
    onBehalfOf          EntityId
                        OPTIONAL,
    challenge           Challeng
                        e,
    number              INTEGER
                        (1..int4)
  }
```

Компонент onBehalfOf, при его наличии, указывает объект, открытый ключ шифрования которого в конечном итоге будет использован для возвращаемых данных. При отсутствии данных указаний значением по умолчанию является объект EntityId регистрирующего органа ТС. Когда ERT еще не введена в эксплуатацию, значение компонента onBehalfOf игнорируется.

Компонент вызова — это случайное число, генерируемое считывателем ERI, которое в конечном итоге должно быть возвращено в подписанной части результата, для того чтобы показать, что возвращенный результат является результатом этого вызова, а не реплицированного сообщения.

Сохраненные аргументы транзакций CommissionErt последовательно нумеруются. Первый сохраненный аргумент пронумерован 1.

Компонент number идентифицирует аргумент вызова CommissionErt.

Когда значением числового компонента является число удаленных аргументов, возвращается самый старый доступный аргумент успешно вызванной транзакции CommissionErt.

Когда значение числового компонента более числа последнего сохраненного аргумента, данные этого последнего аргумента должны быть возвращены.

6.2.12.3 Исторические данные ввода в эксплуатацию

Тип HistoricComData используется для исторических данных ввода в эксплуатацию и определяется следующим образом:

```
HistoricComData ::= SEQUENCE {
    number              INTEGER (1..int4),
    outOf               INTEGER (1..int4),
    historicRecord      CommissionErtArgument
    }
```

Компонент number идентифицирует аргумент транзакции CommissionErt, переданный в компоненте historicRecord.

Компонент outOf указывает номер последнего хранимого аргумента транзакции CommissionErt.

Если ERT содержит только текущие данные ввода в эксплуатацию и не содержит исторических данных, значение как числа, так и компонентов, не входящих в число, должно быть 1.

Компонент historicRecord содержит аргумент успешного вызова транзакции CommissionErt, идентифицированного значением числового компонента.

6.2.13 Получение исторических данных ввода в открытый доступ

6.2.13.1 Передача данных о транзакциях с чистотой

Операция getCleartextHistoricComData используется для извлечения аргумента предыдущих транзакций CommissionErt в открытом тексте и определяется следующим образом:

```
getCleartextHistoricComData TRANSACTION ::= {
    ARGUMENT            GetCleartextHistoricComDataAr
                        gument
    RESULT              CLEAR-SECURED
                        {HistoricComData}
    ERRORS              {notCommissioned}
    CODE                13
     }
```

Поле ARGUMENT имеет тип GetCleartextHistoricComDataArgument.

Сделка возвращает исторические данные ввода в эксплуатацию, подписанные или неподписанные, в соответствии с возможностями ERT (см. 6.2.18.1 для определения CLEAR-SECURED {}).

Если ERT еще не введена в эксплуатацию или если ввод в эксплуатацию снят, возвращается код ошибки notCommissioned.

6.2.13.2 Аргумент транзакции с данными о первоначальном вводе данных с открытым текстом

Тип GetCleartextHistoricEriDataArgument определен следующим образом:

```
GetCleartextHistoricComDataArgument ::= SEQUENCE {
    credentials         ErtHolderCreden
                        tials,
    challenge           Challenge,
    number              INTEGER
                        (1..int4)
}
```

Компоненты GetCleartextHistoricComDataArgument определяются как в GetCleartextHistoricEriData Argument.

6.2.14 Обновление списка управления доступом

6.2.14.1 Переменная списка управления доступом к обновлению

Транзакция updateAccessControlList должна использоваться для добавления или удаления записей из списка управления доступом ERT. Транзакция updateAccessControlList определена следующим образом:

```
updateAccessControlList TRANSACTION ::= {
    ARGUMENT            UpdateAccessControlListAr
                        gument
    RESULT              NULL
    ERRORS              {UpdateAccessControlListE
                        rrors}
    CODE                11
}
```

Операция updateAccessControlList вызывается с аргументом типа updateAccessControlListArgument.

Сделка может быть вызвана либо регистрирующим органом, который вызвал ERT, либо владельцем ERT.

Если транзакция выполнена успешно, ERT возвращает значение NULL. В противном случае возвращается один из кодов ошибок из UpdateAccessControlListErrors.

Данные не должны храниться в ERT, если транзакция не выполнена успешно.

6.2.14.2 Аргумент списка управления доступом к обновлению

Тип UpdateAccessControlListArgument определен следующим образом:

```
UpdateAccessControlListArgument ::= CHOICE {
        authorityUpdate   SIGNED {AccessControlListUpdate},
        holderUpdate      HolderAccessControlListUpdate
        }
```

Альтернатива authorityUpdate используется, когда список управления доступом ERT обновляется регистрирующим органом. Альтернатива владельца Update должна использоваться при обновлении списка контроля доступа владельцем ERT.

Компонент authorityUpdate, при его наличии, должен иметь данные AccessControlListUpdate, подписанные с таким же ключом, который использовался для последней успешно выполненной транзакции ввода в эксплуатацию. В противном случае возвращается код ошибки незаконной системы.

При наличии у ERT возможности проверки подписи подпись компонента authorityUpdate должна быть проверена с сертификатами, включенными в аргумент последней успешно выполненной транзакции ввода в эксплуатацию.

6.2.14.3 Тип обновления списка контроля доступа держателя

Тип HolderAccessControlListUpdate определен следующим образом:

```
HolderAccessControlListUpdate ::= SEQUENCE {
        credentials                ErtHolderCredentials,
        entry              AccessControlListUpdate
        }
```

Компонент учетных данных должен содержать учетные данные (vehicleId плюс PIN) держателя ERT. Учетные данные проверяются ERT.

6.2.14.4 Тип обновления списка управления доступом

Тип AccessControlListUpdate используется для указания обновления списка управления доступом ERT и определяется следующим образом:

```
AccessControlListUpdate ::= SEQUENCE {
    mode                    ENUMERATED {deleteAllAndAdd (0),
                        add (1), delete (2)}
                            DEFAULT add,
    entry                   AccessControlEntry OPTIONAL
  }
```

Компонент режима определяет режим обновления и может иметь одно из следующих значений:

- deleteAllAndAdd — в этом случае сначала либо все записи, добавленные регистрирующим органом, удаляются, если транзакция вызывается регистрирующим органом, либо все записи, добавленные владельцем ERT, удаляются, если транзакция вызывается держателем ERT. Затем добавляется новая запись, если она присутствует;
- добавления — для того, чтобы добавить новую запись в список управления доступом. Записи, внесенные регистрирующим органом, добавляются в качестве органов. Записи, внесенные владельцем ERT, добавляются в качестве дополнительных поставщиков услуг;
- удаления — для того, чтобы удалить запись из списка управления доступом. Регистрирующий орган может удалять только записи;

- добавленного (возможно, другим) регистрирующим органом. Держатель ERT может удалять только записи, добавленные (возможно, предыдущим) держателем ERT.

Элемент записи, при его наличии, должен содержать запись, которую необходимо добавить или удалить.

Компонент записи должен отсутствовать, если компонент режима имеет значение deleteAllAndAdd, и никакие записи не добавляются. В противном случае компонент записи должен присутствовать.

Если режим «добавляет» и если уже есть запись для объекта с той же сущностью, добавленная запись должна полностью заменить старую.

6.2.14.5 Тип элемента управления доступом

Тип AccessControlEntry определен следующим образом:

```
        AccessControlEntry          ::=
SEQUENCE{
                id                          EntityId,
                name                        Text OPTIONAL,
                enciphermentKeyId           KeyId OPTIONAL,
                                            PublicEnciphermentKey OPTIONAL
                publicEnciphermentKey
Reader
                }
```

Компонент id должен идентифицировать объект, права доступа которого добавляются или удаляются.

Компонент имени, при его наличии, должен содержать имя объекта, права доступа которого добавляются или удаляются. Если запись должна быть удалена, любая ценность, присутствующая в ERT, должна быть проигнорирована.

Компонент enciphermentKeyId, при его наличии, идентифицирует открытый ключ шифрования, используемый для шифрования. Различные объекты могут использовать одинаковые идентификаторы.

Примечание — При возврате с зашифрованными данными это значение может быть использовано для определения соответствующего частного ключа дешифрования.

Компонент enciphermentKeyId должен присутствовать при наличии компонента public EnciphermentKeyReader. В противном случае компонент должен отсутствовать.

Компонент publicEnciphermentKeyReader, при его наличии, должен содержать ключ общедоступного шифрования объекта, права доступа которого добавляются. Этот ключ должен использоваться для шифрования данных ERI перед его отправкой в виде зашифрованного текста в считыватель ERI, действующий от имени объекта, идентифицированного компонентом id.

Компонент publicEnciphermentKeyReader должен отсутствовать, если значением компонента режима является «delete». В противном случае компонент должен присутствовать.

6.2.14.6 Ошибки транзакций списка управления обновлениями

Набор значений UpdateAccessControlListErrors определен следующим образом:

```
    UpdateAccessControlListErrors ErrorCode ::= {
            illegalArgument                 |
    illegalVehicleId                        |
    illegalSignature                        |
    illegalHolderAccess   |
            illegalDate |       noEntry |
            resourceLimitExceeded |   noEnciphermentCapability |
            otherError          }
```

Значения UpdateAccessControlListErrors имеют тип EriErrorCode и содержат перечисленные значения.

6.2.15 Получение списка записей контроля зашифрованного текста

6.2.15.1 Операция списка управления доступом к шифротексту

Операция getCiphertextAccessControlListEntry должна быть использована для извлечения записи из списка управления доступом в результате обновления полномочий этого списка в зашифрованном тексте из ERT и определяется следующим образом:

```
getCiphertextAccessControlListEntry TRANSACTION ::= {
        ARGUMENT                GetCiphertextAccessControlListEntryAr
                        gument
        RESULT                  SECURED
                        {AuthorityAccessControlListEntry}
        CODE                    9
          }
```

Поле ARGUMENT имеет тип GetCiphertextAccessControlListEntryArgument.

Транзакция возвращает данные AuthorityAccessControlListEntry, подписанные и/или зашифрованные в соответствии с возможностями ERT (см. 6.2.18.1 для определения SECURED {}).

При шифровании ключ общедоступного шифрования — это ключ общедоступного шифрования регистрирующего органа, который заказал ERT.

Примечание — Идентификатор регистрирующего органа, который ввел ERT, можно получить с помощью транзакции getEriData.

Операция getCiphertextAccessControlListEntry работает в списке управления доступом. Этот список контроля доступа представляет собой список всех записей управления доступом в ERT, которые добавлены или обновлены с помощью транзакции updateAccessControlList с аргументом полномочий Update. Каждой записи в этом списке, при ее наличии, задается отдельный номер между 1 и количеством записей в списке.

Список контроля доступа может быть пустым.

6.2.15.2 Аргумент транзакции ввода списка доступа к шифруемому типу

Тип GetCiphertextAccessControlListEntryArgument определен следующим образом:

```
GetCiphertextAccessControlListEntryArgument ::= SEQUENCE {
        challenge       Challenge,
        number          INTEGER (1..int4)
        }
```

Компонент вызова — это случайное число, генерируемое считывателем ERI, которое в конечном итоге должно быть возвращено в подписанной части результата, для того чтобы показать, что возвращенный результат является результатом этого вызова, а не реплицированного сообщения.

Компонент number указывает номер записи, которая будет извлечена из списка управления доступом, на котором работает транзакция.

6.2.15.3 Права для доступа к контрольному списку записи

Типы AuthorityAccessControlListEntry и AccessControlListEntry определены следующим образом:

```
        AuthorityAccessControlListEntry     ::=
AccessControlListEntry (WITH COMPONENT {...,
holderEntry ABSENT}     )

        AccessControlListEntry    ::=       SEQUENCE {
        number        INTEGER        (0..int4),          outOf
INTEGER        (0..int4),
            authorityEntry
```

```
    AccessControlEntry OPTIONAL,
        holderEntry
    AccessControlEntry OPTIONAL
        }
```

Для типа AccessControlListEntry отсутствует запись о владельце ТС.

Компонент number должен указывать номер найденной записи в списке управления доступом, в котором работает транзакция. Если этот список пуст, его значение равно 0.

Когда значение компонента числа в аргументе транзакции было менее или равно количеству записей в списке управления доступом, на котором работает транзакция, значение компонента числа должно быть таким же, как и значение компонента числа в аргументе, то есть запись, указанная в аргументе, будет восстановлена.

Если значение числового компонента в аргументе транзакции более, чем количество записей в непустом списке управления доступом, на котором работает транзакция, значение числового компонента должно быть равно количеству записей в списке, т. е. будет восстановлена последняя запись в списке.

Компонент outOf должен содержать количество записей в списке контроля доступа, в котором работает транзакция.

Компонент authorityEntry, при его наличии, указывает запись в списке управления доступом, имеющем значение компонента числа.

Власть Entry должна присутствовать только тогда, когда запись в списке управления доступом, на которой работает транзакция, является записью, добавляемой или обновляемой посредством транзакции updateAccessControlList с аргументом authorityUpdate.

Компонент holderEntry, при его наличии, указывает запись в списке управления доступом, имеющем значение компонента числа.

Компонент holderEntry должен присутствовать только тогда, когда запись в списке управления доступом, на котором работает транзакция, является записью, добавляемой или обновляемой посредством транзакции updateAccessControlList с аргументом ownerUpdate.

6.2.16 Получение списка контроля доступа в открытом доступе

6.2.16.1 Операция ввода списка доступа к открытому контексту

Операция getCleartextAccessControlListEntry должна использоваться владельцем ERT или от ее имени для получения записи открытого текста из списка управления доступом, содержащегося в ERT. Операция getCleartextAccessControlListEntry определена следующим образом:

```
getCleartextAccessControlListEntry TRANSACTION ::= {
    ARGUMENT        GetCleartextAccessControlListEntryArgument
    RESULT          CLEAR-SECURED {AccessControlListEntry}
    ERRORS          {GetCleartextAccessControlListEntryErrors}
    CODE            10
      }
```

Поле ARGUMENT имеет тип GetCleartextAccessControlListEntryResult.

Если транзакция выполнена успешно, ERT возвращает значение CLEAR-SECURED {AccessControlListEntry}. В противном случае возвращается один из кодов ошибок из GetCleartextAccessControlListEntryErrors.

Значение CLEAR-SECURED {AccessControlListEntry} содержит данные AccessControlListEntry, подписанные или неподписанные, в соответствии с возможностями ERT (см. 6.2.18.1 для определения CLEAR-SECURED {}).

Транзакция работает со списком всех записей управления доступом, содержащихся в ERT. ERT присваивает каждой записи в этом списке, при ее наличии, отдельное число от 1 до количества записей в этом списке.

6.2.16.2 Сценарий транзакции ввода списка доступа для открытого текста

Функция GetCleartextAccessControlListEntryArgument определена следующим образом:

```
GetCleartextAccessControlListEntryArgument ::= SEQUENCE {
        credentials                 ErtHolderCreden
                                tials,
        challenge                   Challenge,
        number                      INTEGER
                                (1..int4)
    }
```

Компонент учетных данных должен содержать учетные данные (VehicleId и PIN) держателя ERT.

Компонент вызова — это случайное число, генерируемое считывателем ERI, которое в конечном итоге должно быть возвращено в подписанной части результата, для того чтобы показать, что возвращенный результат является результатом этого вызова, а не реплицированного сообщения.

Компонент number указывает номер записи, которая будет извлечена из списка управления доступом, на котором работает транзакция.

6.2.16.3 Ошибки транзакции входа в список очистки открытого текста

Набор значений GetCleartextAccessControlListEntryErrors определен следующим образом:

```
GetCleartextAccessControlListEntryErrors ErrorCode ::= {
        illegalArgument | illegalVehicleId |
        illegalHolderAccess |       otherError
        }
```

Тип GetCleartextAccessControlListEntryErrors имеет тип ErrorCode и должен принимать одно из перечисленных значений.

6.2.17 Получение возможности ERT

6.2.17.1 Операция получения возможностей ERT

Транзакция getErtCapabilities должна использоваться для получения возможностей ERT и определяется следующим образом:

```
getErtCapabilities TRANSACTION ::= {
        ARGUMENT                    NULL
        RESULT                      ErtCapabilities
        CODE                        15
        }
```

Поле аргумента имеет тип NULL.

Поле RESULT имеет тип ErtCapabilities, как определено ниже.

6.2.17.2 Результат транзакции получения возможностей ERT

Тип ErtCapabilities определен следующим образом:

```
        ErtCapabilities ::= SEQUENCE
{
            supportedTransactions           SEQUENCE OF INTEGER,
            hashingAlgorithms               SEQUENCE OF HashingAlgorithm
                                        OPTIONAL,
            signingAlgorithms               SEQUENCE OF PublicKeyAlgorithm
                                        OPTIONAL,
                                            SEQUENCE OF PublicKeyAlgorithm
signatureVerificationAlgorithms         OPTIONAL,
```

```
                                        SEQUENCE OF SecretKeyAlgorithm
secretKeyEncryptionAlgorithms           OPTIONAL,
                                        SEQUENCE OF PublicKeyAlgorithm
publicKeyEncryptionAlgorithms           OPTIONAL,
                                        SEQUENCE OF PublicKeyAlgorithm
publicKeyDecryptionAlgorithms           OPTIONAL,
        maxBitsPublicKeys               INTEGER (0..int4),
        maxOctetsPin                    INTEGER (0..int4),
        maxOctetsSetArgument            INTEGER (0..int4),
        maxNumberSetArguments           INTEGER (1..int4),
        maxNumberComArguments           INTEGER (1..int4),
        maxSizeAccessControlList        INTEGER (0..int4),
        maxNumberAuthorities            INTEGER (0..int4),
                                        INTEGER (0..int4),
maxNumberAddServProviders
                                        ErtSecurityFlags,
 ertSecurityIndicationSupport
        maxInteger                      INTEGER (1..int4),
        maxStringSize                   INTEGER (1..int4),
        ...
        }
```

Примечание — Необходимость указывать, отключено ли регистрационное агентство для шифрования и аутентификации, отсутствует. Можно вызвать транзакцию получения ERI-данных и посмотреть на результат. Регистрирующий орган может также определить это, проверив аргумент последней операции по вводу в эксплуатацию, используя транзакцию передачи данных зашифрованного ввода.

Компонент supportedTransactions указывает поддерживаемые транзакции, идентифицированные значением поля кода транзакции.

Компонент hashingAlgorithms, при его наличии, указывает поддерживаемые алгоритмы хеширования.

Компонент hashingAlgorithms должен присутствовать, если ERT имеет возможности аутентификации или подписи. В противном случае он отсутствует.

Компонент signingAlgorithms, при его наличии, указывает поддерживаемые алгоритмы открытого ключа для подписания.

Компонент signingAlgorithms должен присутствовать при наличии у ERT возможности подписи. В противном случае он отсутствует.

Компонент signatureVerificationAlgorithms, при его наличии, указывает поддерживаемые алгоритмы открытых ключей для проверки подписи.

Компонент signatureVerificationAlgorithms должен присутствовать, если ERT имеет возможности проверки подписи. В противном случае он отсутствует.

Компонент secretKeyEncryptionAlgorithms, при его наличии, указывает поддерживаемые алгоритмы секретных ключей.

Компонент secretKeyEncryptionAlgorithms должен присутствовать, если ERT имеет возможности шифрования. В противном случае он отсутствует.

Компонент publicKeyEncryptionAlgorithms, при его наличии, указывает поддерживаемые алгоритмы открытого ключа для шифрования.

Компонент publicKeyEncryptionAlgorithms должен присутствовать, если ERT имеет возможности шифрования. В противном случае он отсутствует.

Компонент publicKeyDecryptionAlgorithms, при его наличии, указывает поддерживаемые алгоритмы открытого ключа для дешифрования.

Компонент publicKeyDecryptionAlgorithms должен присутствовать, если ERT имеет возможности дешифрования. В противном случае он отсутствует.

Компонент maxBitsPublicKeys указывает максимальное количество битов, которое может использоваться для открытого или закрытого ключа.

Если алгоритмы открытых ключей не поддерживаются, значение компонента maxBitsPublicKeys равно нулю.

Компонент maxOctetsPin указывает максимальное количество символов, которое может использоваться для PIN-кода.

Если PIN-коды не поддерживаются, значение компонента maxOctetsPin равно нулю.

Компонент maxOctetsSetArgument указывает максимальное количество октетов, которые могут использоваться для аргумента транзакции данных ERI.

Компонент maxNumberSetArguments указывает максимальное количество аргументов заданной транзакции данных ERI, которые могут быть сохранены в ERT.

Компонент maxNumberComArguments указывает максимальное количество аргументов транзакции ввода в эксплуатацию, которые могут быть сохранены в ERT.

Компонент maxSizeAccessControlList определяет максимальное количество записей в списке управления доступом в ERT.

Компонент maxNumberAuthorities указывает максимальное количество записей для полномочий в списке управления доступом в ERT. Это число должно быть менее или равно значению компонента maxSizeAccessControlList.

Компонент maxNumberAddServProviders указывает максимальное количество записей для дополнительных поставщиков услуг в списке управления доступом в ERT. Это число должно быть менее или равно значению компонента maxSizeAccessControlList.

Сумма значений maxNumberAuthorities и компонентов maxNumberAddServProviders может быть более чем значение компонента maxSizeAccessControlList.

Компонент ertSecurityIndicationSupport указывает поддержку возможностей безопасности ERT и имеет тип ErtSecurityFlags. Бит «1» указывает, что поддерживается соответствующая возможность, бит «0» — не поддерживается.

Примечание — Такой же тип используется для компонента ertSecurityStatus в типе CleartextEriData, но с другим значением, прикрепленным к битам.

Компонент maxInteger задает максимальное значение целого числа, которое может обрабатывать ERT.

Компонент maxStringSize указывает максимальное количество октетов, которые могут использоваться для строки в этой ERT.

6.2.18 Общие определения

6.2.18.1 Данные с отметкой и защитой ERT

а) Защищенные данные

Параметрированный тип SECURED должен использоваться ERT для защиты данных в соответствии с его возможностями и определяется следующим образом:

```
        SECURED         {ToBeSecured}
::= CHOICE {
                                                ENCRYPTED {ERT-AUTHENTICATED {TAGGED
        authenticatedAndEncrypte        {ToBeSecured}}},
d
        authenticated                           ERT-AUTHENTICATED               {TAGGED
                                        {ToBeSecured}},
        encrypted                               ENCRYPTED {TAGGED {ToBeSecured}},
        cleartext                               TAGGED {ToBeSecured}
        }
```

Для всех альтернатив параметрированный тип TAGGED добавляет уникальный номер ERT и, в случае применения, вызов и порядковый номер (см. определение параметрированного типа TAGGED ниже).

Аутентифицированная альтернатива AndEncrypted должна использоваться, если ERT введена в эксплуатацию и имеет как включенное шифрование, так и возможности подписи.

Аутентифицированная альтернатива AndEncrypted, при ее наличии, должна содержать тегированные данные TAGGED {ToBeSecured}, прошедшие проверку подлинности и впоследствии зашифрованные ERT.

Аутентифицированная альтернатива должна использоваться, если ERT вводится в эксплуатацию и имеет возможность подписи, но не шифрования.

Аутентифицированная альтернатива, при ее наличии, должна содержать тегированные данные TAGGED {ToBeSecured}, аутентифицированные ERT.

Зашифрованная альтернатива должна использоваться, если ERT вводится в эксплуатацию и имеет возможности шифрования, но не подписи.

Зашифрованная альтернатива, при ее наличии, должна содержать тегированные данные TAGGED {ToBeSecured}, зашифрованные ERT.

Альтернатива cleartext должна использоваться, если ERT не введена в эксплуатацию, не имеет возможности включения шифрования и не поддерживает возможности подписи.

Альтернатива открытого текста, при ее наличии, содержит помеченные данные TAGGED {ToBeSecured}.

Ключ общего шифрования, используемый для шифрования, должен быть ключом, идентифицированным в компоненте onBehalfOf в аргументе вызываемой транзакции, или, если не доступен, ключом общедоступного шифрования регистрирующего органа, который заказал ERT.

Ключ частной подписи, используемый для аутентификации, должен быть указан в последней транзакции ввода в эксплуатацию.

б) Чистые защищенные данные

Параметрированный тип CLEAR-SECURED должен использоваться ERT для защиты данных открытого текста в соответствии со своими возможностями и определяется следующим образом:

```
CLEAR-SECURED {ToBeSecured} ::= SECURED {ToBeSecured} (WITH
COMPONENTS
        {authenticatedAndEncrypted ABSENT, encrypted ABSENT} )
```

в) Маркированные данные

Параметрированный тип TAGGED должен использоваться для добавления номера ERT и данных конкретной транзакции к результату транзакции чтения до того, как этот результат будет подписан или зашифрован, если допустимо. Параметрированный тип TAGGED определен следующим образом:

```
TAGGED {ToBeTagged} ::= SEQUENCE {
        ertNumber               ErtNumber,
        challenge               Challenge
                                OPTIONAL,
        sequenceNumber          INTEGER (1..int4)
                                OPTIONAL,
        tobeTagged              ToBeTagged
}
```

Компонент ertNumber должен содержать уникальный номер ERT.

Компонент вызова, при его наличии, должен содержать значение параметра вызова из аргумента вызываемой транзакции.

Компонент запроса должен присутствовать при выполнении обоих условий: ERT активировала возможности подписи, и проблема присутствовала в аргументе вызова транзакции.

Компонент вызова может присутствовать, когда ERT не имеет разрешенных возможностей подписи, но проблема присутствует в аргументе вызова транзакции.

Компонент запроса не должен присутствовать в случае его отсутствия в аргументе вызова транзакции.

Примечание — Компонент запроса гарантирует, что подписанные данные не являются реплицированным сообщением, а действительно результатом этого конкретного вызова транзакции.

Компонент sequenceNumber, при его наличии, должен содержать порядковый номер транзакции чтения. Это число должно начинаться с 1 и должно быть увеличено на 1 после использования в результате транзакции.

Если номер последовательности достигает своего максимального значения, его следующее значение равно 1.

Компонент sequenceNumber должен присутствовать, когда ERT активировала возможности шифрования.

Компонент sequenceNumber может присутствовать или не присутствовать, когда ERT не имеет разрешенных возможностей шифрования.

Примечание — Этот порядковый номер гарантирует, что два зашифрованных сообщения ERT не совпадают, даже если используется одна и та же проблема. Это предотвращает при зашифровании появление «псевдонимов», т. е. сообщений, которые непреднамеренно идентифицируют ТС только потому, что сообщения одинаковы.

6.2.18.2 Определения аутентификации ERT

а) Аутентификация ERT

Параметрированный тип ERT-AUTHENTICATED определен следующим образом:

```
ERT-AUTHENTICATED {ToBeErtAuthenticated} ::= SEQUENCE {
        ertSigned           SIGNED {ToBeErtAuthenticated, PrivateSignatureKey},
publicVerificationKeyCertificate      ErtCertificate
        }
```

Компонент ertSigned содержит аутентифицированные данные, т. е. данные, подписанные ключом секретной подписи ERT.

Компонент publicVerificationKeyCertificate должен содержать сертификат открытого ключа, выданный регистрирующим органом ТС, для открытого ключа проверки, который будет использоваться для проверки подписей ERT.

Примечание — Этот сертификат открытого ключа должен быть записан в ERT как часть транзакции комиссии ERT.

б) Сертификат ERT

Тип ErtCertificate определен следующим образом:

```
ErtCertificate ::= SIGNED {ErtCertificationData, PrivateSignatureKey}
```

ErtCertificationData должен иметь содержимое сертификата ERT.

ErtCertificationData подписывается с ключом частной подписи регистрирующего органа.

Примечания

1 Не требуется, чтобы алгоритм хеширования и алгоритм открытого ключа поддерживались ERT. Сертификат ERT предоставляется регистрирующим органом в транзакции комиссии ERT и возвращается ERT в транзакции чтения. Сертификат ERT не контролируется и не интерпретируется ERT.

2 Тем не менее, так как читатель ERI должен иметь возможность использовать сертификаты, выданные регистрирующим органом, как алгоритм хеширования, так и алгоритм с открытым ключом должны поддерживаться в соответствии с настоящим стандартом.

в) Данные сертификата ERT

Тип данных ErtCertificationData должен использоваться для сертифицируемых данных и определяется следующим образом:

```
        ErtCertificationData
::= SEQUENCE {
                vehicleId                   VehicleId,
```

```
                                    PublicVerificationKey,
    publicVerificationKey
        signatoryId                 EntityId,
        date                        DATE
    }
```

Компонент vehicleId содержит файл vehicleId.

Компонент publicVerificationKey содержит открытый ключ проверки для проверки подписей ERT.

SignatoryId содержит идентификатор регистрирующего органа, выдавшего сертификат.

Компонент даты содержит дату подписания сертификата.

6.2.18.3 Определения аутентификации BOE

а) Параметрированный тип DATED используется для прикрепления данных достоверности и определяется следующим образом:

```
DATED {ToBeDated} ::= SEQUENCE {
        date                        DATE,
        validThru                   DATE OPTIONAL,
        issuer                      EntityId,
        toBeDated                   ToBeDated
        }
```

Компонент данных должен содержать дату, фиксирующую подпись значения параметра ToBeDated.

Значение параметра ToBeDated становится действительным в момент, указанный в компоненте даты.

Компонент validThru, при его наличии, указывает ту дату, до истечения которой действительна подпись для значения параметра ToBeDated.

Когда компонент validThru отсутствует, значение параметра ToBeDated действует в течение неограниченного периода времени.

Компонент эмитента идентифицирует значение параметра ToBeDated, т. е. производителя или регистрирующего органа, который подписал это значение.

Компонент toBeDated содержит значение параметра ToBeDated.

б) Проверка подлинности BOE

Параметрированный тип BOE-AUTHENTICATED используется для аутентификации данных, выданных регистрирующим органом, и определяется следующим образом:

```
        BOE-AUTHENTICATED {ToBeBoeAuthenticated} ::= SEQUENCE {
            signedParameter SIGNED {ToBeBoeAuthenticated, PrivateSignatureKey},
    certificates SEQUENCE SIZE(1..2) OF BoeCertificate
                    }
```

Компонент signedParameter должен содержать значение параметра ToBeBoeAuthenticated, подписанного с помощью ключа частной подписи регистрирующего органа.

Компонент сертификатов должен содержать последовательность из одного или двух сертификатов открытых ключей типа BoeCertificate, как определено ниже.

Когда компонент сертификатов содержит только один сертификат открытого ключа, этот сертификат должен выдаваться центром сертификации верхнего уровня для ключа открытой подписи регистрирующего органа, данные которого должны быть аутентифицированы.

Когда компонент сертификатов содержит два сертификата открытого ключа, первый из них выдается центром сертификации верхнего уровня для ключа общей подписи промежуточного центра сертификации, который подписал второй сертификат. Второй сертификат выдается промежуточным центром сертификации для открытого ключа подписи регистрирующего органа, данные которого должны быть аутентифицированы.

Сертификаты открытого ключа в последовательности должны быть действительными на дату подписания значения параметра ToBeBoeAuthenticated.

в) Сертификат BOE

Тип BoeCertificate определяется следующим образом:

```
BoeCertificate ::= SIGNED {BoeCertificationData, PrivateSignatureKey}
```

BoeCertificationData должен содержать BoeCertificationData.

BoeCertificationData подписывается центром сертификации с использованием алгоритма хеширования и алгоритма открытого ключа, поддерживаемого ERT.

Примечания

1 ERT должна проверить подпись регистрирующего органа, используя один или несколько сертификатов BOE.

2 Список поддерживаемых алгоритмов можно получить, вызвав транзакцию get ERT.

г) Данные сертификации BOE

Тип CertificationData определен следующим образом:

```
        BoeCertificationData
::= SEQUENCE {
            entityId                    EntityId,
            entityRole                  EntityRole,
            entityName                  Text OPTIONAL,
            publicKey                   Key,
            signatoryId                 EntityId,
            signatoryName               Text OPTIONAL,
            signatoryRole               EntityRole,
            date                        DATE,
            validThru                   DATE
            }
```

Компонент entitityId идентифицирует сертифицированный объект.

Компонент entityRole определяет роль сертифицированного объекта.

Компонент entityName, при его наличии, содержит имя сертифицированного объекта на языке, выбранном подписью сертификата.

Компонент publicKey содержит открытый ключ сертифицированного объекта.

Компонент signatoryId идентифицирует подписчика сертификата.

Компонент signatoryName, при его наличии, должен содержать имя подписавшего на выбранном им языке.

Компонент signatoryRole определяет роль подписавшего.

Компонент даты содержит дату подписания сертификата.

Элемент validThru указывает ту дату, до истечения которой действителен сертификат.

д) Роли объекта

Тип EntityRole определяет роль сущности и определяется следующим образом:

```
        EntityRole::= ENUMERATED {
                topLevelCertificationAuthority                                  (0),
intermediateCertificationAuthority (1),
                manufacturer (2),           registrationAuthority (3),
                authority (4),      serviceProvider (5),        eriHolder (6)      }
```

Тип EntityRole может принимать одно из определенных значений.

6.2.18.4 Подписание и подписи

а) Подпись

Параметрированный тип SIGNED должен использоваться для подписи данных и определяется следующим образом:

```
SIGNED {ToBeSigned, PrivateSignatureKey} ::= SEQUENCE {
        toBeSigned          ToBeSigned,
        hashingAlgorithm    HashingAlgorithm DEFAULT sha1,
        signatureAlgorithm  PublicKeyAlgorithm
DEFAULT ellipticCurve,  signature   SIGNATURE
{ToBeSigned, HashingAlgorithm,
                PublicKeyAlgorithm, PrivateSignatureKey}
        }
```

Компонент toBeSigned должен содержать значение параметра SIGNED ToBeSigned.

Параметр PrivateSignatureKey должен содержать значение ключа частной подписи, используемого для подписи компонента toBeSigned.

Компонент hashingAlgorithm, при его наличии, должен указывать алгоритм хеширования, используемый для вычисления хеш-кода по параметру ToBeSigned. В противном случае значением по умолчанию является sha1.

Компонент signatureAlgorithm, если он присутствует, должен указывать алгоритм открытого ключа, используемый для подписи хеш-кода по параметру ToBeSigned, в противном случае значение по умолчанию — эллиптическое.

Компонент подписи должен содержать подпись для значения параметра ToBeSigned с использованием алгоритмов hashingAlgorithm и signatureAlgorithm [см. перечисление б) 6.2.18.4].

б) Проверка подписи

Параметрированный тип SIGNATURE определен следующим образом:

```
SIGNATURE   {ToBeSigned,    HashingAlgorithm,   SignatureAlgorithm,
        PrivateSignatureKey}    ::=     BIT STRING  (CONSTRAINED BY
        {HashingAlgorithm, -- and -- SignatureAlgorithm, -- with --
PrivateSignatureKey, -- applied to -- ToBeSigned} )
```

Значение SIGNATURE — это BIT STRING, которая является результатом двухэтапного процесса. Во-первых, хеш-код вычисляется с помощью алгоритма хеширования по закодированному значению параметра ToBeSigned. Затем этот хеш-код зашифровывается с помощью алгоритма подписи с открытым ключом, используя значение параметра PrivateSignatureKey в качестве ключа частной подписи.

в) Параметр вызова

Тип задачи определен следующим образом:

```
        Challenge ::= INTEGER (1..int4)
```

6.2.18.5 Общие определения шифрования

а) Шифрование ERT

Параметрированный тип ENCRYPTED определен следующим образом:

```
ENCRYPTED {ToBeErtEncrypted}    ::=     SEQUENCE {
        enciphermentKeyId       KeyId,
        publicKeyEncryptionAlgorithm    PublicKeyAlgorithm DEFAULT ellipticCurve,
secretKeyEncryptionAlgorithm    SecretKeyAlgorithm DEFAULT aes,
        encryptedKey    KEY-ENCRYPTION {SecretTransactionKey,       PublicKeyAlgorithm,
PublicEnciphermentKey},  ciphertext     CIPHER-TEXT {ToBeErtEncrypted,      SecretKeyAlgorithm,
SecretTransactionKey}
        }
```

Компонент enciphermentKeyId идентифицирует ключ общедоступного шифрования, используемый для шифрования секретного ключа транзакции. Его значение должно использоваться для определения соответствующего частного ключа дешифрования.

Если шифрование выполняется с помощью ERT с включенными возможностями шифрования и отсутствует открытый ключ шифрования, ключ Id должен принимать произвольное значение.

Примечание — В обычных условиях ERT никогда не должна использовать случайное значение для keyId. Однако, если ключ не доступен, случайное значение затрудняет для нелегального читателя возможность заметить, что ключ не доступен.

Компонент publicKeyEncryptionAlgorithm, при его наличии, должен идентифицировать алгоритм открытого ключа, используемый для шифрования случайно выбранного секретного ключа для шифрования данных ToBeEncrypted. В противном случае значением по умолчанию является алгоритм эллиптической кривой.

Компонент secretKeyEncryptionAlgorithm, при его наличии, должен идентифицировать алгоритм секретного ключа, используемый для шифрования данных ToBeEncrypted. В противном случае значением по умолчанию является алгоритм AES.

Компонент encryptedKey содержит секретный случайный ключ транзакции, зашифрованный с помощью общедоступного ключа шифрования.

Примечание — В критически важных ситуациях ERT может зашифровать этот секретный ключ транзакции до того, как будет вызвана транзакция, для которой она необходима.

Компонент зашифрованного текста содержит результат шифрования параметра ToBeEncrypted секретным ключом транзакции.

Примечание — По соображениям производительности данные ERI зашифровываются симметричным ключом только один раз, который зашифрован с помощью асимметричного открытого ключа шифрования.

б) Шифрование с открытым ключом

Параметрированный тип KEY-ENCRYPTION определен следующим образом:

```
KEY-ENCRYPTION {KeyToBeEncrypted, PublicKeyAlgorithm, PublicEnciphermentKey} ::=
        BIT STRING ( CONSTRAINED BY {
        PublicKeyAlgorithm, -- with -- PublicEnciphermentKey, -- applied to -- KeyToBeEncrypted -- or random bit string with same length if no public key is available --} )
```

Значение BIT STRING является результатом шифрования открытого ключа значения параметра ToBeEncrypted с использованием алгоритма открытого ключа, идентифицированного параметром PublicKeyAlgorithm, и значения параметра PublicEnciphermentKey в качестве открытого ключа шифрования.

Если шифрование выполняется ERT с включенными возможностями шифрования и открытым ключом шифрования, BIT STRING принимает случайное значение такой же длины.

в) Шифрование секретного ключа

Параметрированный тип CIPHER-TEXT определен следующим образом:

```
CIPHER-TEXT {ToBeEncrypted, SecretKeyAlgorithm, SecretKey} ::=
        BIT STRING ( CONSTRAINED BY {
        SecretKeyAlgorithm, -- with -- SecretKey, -- applied to -- ToBeEncrypted -- or random bit string with same length if no public key is available --} )
```

Значение BIT STRING является результатом шифрования значения параметра ToBeEncrypted с использованием алгоритма секретного ключа, идентифицированного параметром SecretKeyAlgorithm, и значения параметра SecretKey в качестве (симметричного) ключа секретного шифрования.

Если шифрование выполняется ERT с включенными возможностями шифрования и открытым ключом шифрования, BIT STRING принимает случайное значение такой же длины.

6.2.18.6 Другие общие определения

а) Удостоверения владельца ERT

Тип ErtHolderCredentials определен следующим образом:

```
ErtHolderCredentials        ::=
SEQUENCE {
        vehicleId                       VehicleId,
        pin                             PIN
        }
```

Компонент vehicleId содержит VehicleId ТС.
Контакт компонента содержит PIN-код для держателя ERT.
б) Криптографические алгоритмы
1) Хеширование алгоритмов
Набор поддерживаемых алгоритмов хеширования определен следующим образом:

```
        HashingAlgorithm ::= ENUMERATED {
                sha1,
                … }
```

В настоящее время поддерживается только алгоритм безопасного шифрования 1 (см. [7]).

Примечание — Определение предоставляется только для поддержки будущих расширений настоящего стандарта.

2) Алгоритмы открытого ключа
Набор поддерживаемых алгоритмов открытых ключей определен следующим образом:

```
        PublicKeyAlgorithm          ::=
ENUMERATED                          {
        ellipticCurve,
                … }
```

В настоящее время поддерживается только алгоритм эллиптической кривой (см. [8] и [9]).

Примечание — Определение предоставляется только для поддержки будущих расширений настоящего стандарта.

3) Алгоритмы секретного ключа
Набор поддерживаемых алгоритмов секретных ключей определен следующим образом:

```
        SecretKeyAlgorithm      ::=   ENUMERATED   {
        aes,
        … }
```

В настоящее время поддерживается только алгоритм AES (см. [10]).

Примечание — Определение предоставляется только для поддержки будущих расширений настоящего стандарта.

в) Ключи и ключевые идентификаторы
Различные типы ключей определены следующим образом:

```
SecretTransactionKey                ::= Key
PublicEnciphermentKey               ::= Key
PrivateDeciphermentKey              ::= Key
```

```
PrivateSignatureKey          ::= Key
PublicVerificationKey        ::= Key
Key                          ::= BIT STRING
PIN                          ::= NumericString (SIZE(4))
KeyId                        ::= INTEGER (0..65535)
```

Тип KeyId имеет тип integer и должен использоваться для идентификации ключа. Он может принимать любое значение от 0 до 65535.

г) Номер ERT

Тип ErtNumber определен следующим образом:

```
ErtNumber ::= INTEGER
```

д) Текст

Тип Text используется для текстового описания на языке, выбранном пользователем, и определяется следующим образом:

```
Text ::= UTF8String (SIZE (1..256))
```

6.2.18.7 Int4

Значение int4 определено следующим образом:

```
int4 INTEGER ::= 4294967295
```

а) Коды ошибок ERI

Тип EriErrorCodes определен следующим образом:

```
ErrorCode ::= ENUMERATED {
    illegalArgument,          illegalVehicleId,
    illegalCertificate,       illegalSignature,
    illegalEntity,            illegalHolderAccess,
    illegalDate,              notCustomized,
    notCommissioned,          noEntry,
        resourceLimitExceeded,
    -- encipherment support errors
        noEnciphermentCapability,    noDeciphermentCapability,
    secretKeyEncryptionAlgorithmNotSupported,
        publicKeyEncryptionAlgorithmNotSupported,
    -- authentication support errors
        noSigningCapability,  noSignatureVerificationCapability,
    hashingAlgorithmNotSupported,
        signingAlgorithmNotSupported,

        otherError,
        ... }
```

Тип EriErrorCode может принимать одно из следующих значений:

- незаконный аргумент — если аргумент имеет незаконную структуру или незаконную ценность;

- illegalVehicleId — если VIN, при его наличии, не соответствует нормативному документу, приведенному в [1], или vehicleId не содержит требуемого значения;
- незаконный сертификат — если один или несколько сертификатов являются незаконными или не содержат правильных значений;
- незаконная подпись — если подпись неверна;
- illEntity — если роль entityId или entity неверна;
- illegalHolderAccess — если учетные данные держателя ERT неверные;
- незаконная дата — если дата была неправильной, например: указана не в течение срока действия сертификата(ов) или не позднее, чем в предыдущих данных аутентификации;
- notCustomized — если ERT еще не настроена (и еще не содержит идентификатор ТС);
- notCommissioned — если ERT настроена, но не введена в эксплуатацию;
- noEntry — если запись отсутствует при наличии идентификатора, например отсутствует запись с определенным идентификатором записи;
- resourceLimitExceeded — если ERT не может хранить данные в аргументе транзакции, так как размер данных превышает его емкость;
- noEncipherментCapability — если для транзакции требуются возможности шифрования, а ERT не имеет возможности шифрования;
- noDecipherментCapability — если для транзакции требуются возможности дешифрования, а ERT не имеет данных возможностей;
- secretKeyEncryptionAlgorithmNotSupported — если транзакция указывает алгоритм шифрования секретного ключа, который не поддерживается ERT;
- publicKeyEncryptionAlgorithmNotSupported — если транзакция указывает алгоритм открытого ключа для шифрования/дешифрования, который не поддерживается ERT;
- noSigningCapability — если для транзакции требуются возможности подписи, в то время как ERT не имеет разрешенной возможности подписи;
- noSignatureVerificationCapability — если для транзакции требуются возможности проверки подписи, в то время как ERT не имеет возможности проверки подписи;
- hashingAlgorithmNotSupported — если транзакция указывает алгоритм хеширования, который не поддерживается ERT;
- signedAlgorithmNotSupported — если транзакция указывает алгоритм открытого ключа для подписи или проверки подписи, который не поддерживается ERT;
- OtherError — если ошибка произошла во время попытки транзакции.

6.3 Интерфейсы ERT

6.3.1 Общие требования к интерфейсу ERT

Данные ERI, данные безопасности ERI и номер ERT могут быть доступны только в том виде, в каком указаны в настоящем стандарте.

Блок данных протокола прикладного уровня, подлежащий обмену с ERT, должен быть блоком данных протокола ERI типа EriPdu, т. е. типа EriRequestPdu или типа EriResponsePdu.

Единица данных протокола ERI должна кодироваться в соответствии с каноническим вариантом кодирования (CANONICAL-PER) ALIGNED.

Протоколы нижнего уровня должны соответствовать протоколам международных стандартов.

Примечание — При необходимости блок данных протокола ERI может быть сегментирован и повторно собран.

6.3.2 Интерфейс

Когда интерфейс между ERT и бортовым или внешним считывателем/считывателем ERI должен соответствовать:

- ERT, действующей как PICC (бесконтактная интегральная плата) типа A или B;
- встроенному считывателю/считывателю ERI, действующему как PCD (устройство с бесконтактным соединением, поддерживающее оба типа, A и B).

Блок данных протокола ERI должен быть напрямую передан с использованием поля INF одного или нескольких I-блоков.

Примечание — Следовательно, блок данных протокола ERI не упаковывается в информационные единицы протокола (см. [11]).

Сегментация и повторная сборка блока данных протокола ERI должны выполняться, если требуется, с цепью.

Столкновения между бортовым считывателем или записывающим устройством и внешним (карманным) считывателем или писателем следует избегать.

Примечание — Для карманного считывателя или записывающего устройства это может быть выполнено, если другое бортовое оборудование ERI отключается при выключенном двигателе и/или когда ТС не приводится в движение.

6.3.3 Ближний радиоинтерфейс

6.3.3.1 Общие требования к ближнему радиусу радиоинтерфейса

Воздушный интерфейс ближнего действия должен обмениваться блоками данных протокола ERI, соответствующими 6.3.1.

Встроенное коммуникационное устройство, обеспечивающее доступ к EPT с малой дальностью, должно обеспечивать передачу блоков данных протокола ERI, полученных от одноранговой сети (сотовой сети), в ERT и обратно.

6.3.3.2 Использование протокола уровня приложения DSRC

а) Общие положения

Если протокол уровня приложения DSRC используется для транзакций ERI, то применяется, как указано в этом разделе.

Примечание — Это делает интерфейс ERI DSRC совместимым с другими интерфейсами приложений DSRC, приведенными в [12].

б) Использование службы инициализации DSRC

Если DSRC-связь должна использоваться для транзакций ERI, услуга инициализации должна применяться следующим образом:

- либо компонент mandApplications, либо компонент nonmandApplications для T-PDU initializationrequest (таблица обслуживания Beacon, BST) должен содержать компонент приложения ERI;
- компонент приложений T-PDU с инициализацией-ответом (Vehicle Service Table, VST) должен содержать компонент приложения ERI.

Значение компонента приложения ERI в запросе инициализации или инициализации-ответа должно быть следующим:

- вспомогательный компонент должен иметь значение «идентификация автоматического ТС»;
- элемент eid может быть опущен и, при его наличии, должен быть проигнорирован приложением ERI;
- компонент параметров может быть опущен и, при его наличии, должен быть проигнорирован защищенными функциями связи с использованием асимметричных методов, определенных в настоящем стандарте.

Примечания

1 Обозначение приложения как обязательное или необязательное и его положение в списке приложений выходят за рамки настоящего стандарта. Они влияют только на приоритет приложения ERI по отношению к другим приложениям, указанным в BST.

2 Однако компонент eid и компонент параметров могут использоваться для других приложений, отличных от ERI, AVI.

в) Использование запроса DSRC-действия

Запрос транзакции ERI отправляется из считывателя/записи ERI на встроенный модуль DSRC в качестве запроса следующим образом:

- значение компонента режима должно быть TRUE (так как все транзакции ERI подтверждены);
- значение компонента eid должно быть 0;
- значение компонента actionType должно быть eriTransaction;
- компонент accessCredentials не должен присутствовать;
- значение компонента accessParameter передается как полученное в ERT значение eriRequestPdu;
- компонент iid не должен присутствовать.

Примечание — Запрос-действие имеет тип Action-Request, который определен следующим образом:

```
Action-Request::=SEQUENCE{
mode BOOLEAN,
```

eid Dsrc-EID,
action Type ActionType,
accessCredentials OCTET STRING (SIZE (0..127,...)) OPTIONAL,
actionParameter Container OPTIONAL,
iid Dsrc-EID OPTIONAL
}
(end of note)

г) Использование ответного действия DSRC

Ответ транзакции ERI, полученный от ERT, отправляется встроенным блоком DSRC во внешний считыватель ERI следующим образом:

- значение компонента eid должно быть 0;
- компонент iid не должен присутствовать;
- значение компонента responseParameter, при его наличии, должно быть значением eriResponsePdu, полученным от ERT;
- компонент ret может не использоваться, однако в случае его использования он игнорируется, если представлен eriResponsePdu.

Примечание — Действие-ответ должно быть типа Action-Response, которое определено следующим образом:

Action-Response :: = SEQUENCE {
Заполните BIT STRING (РАЗМЕР (1)),
eid Dsrc-EID,
iid Dsrc-EID ДОПОЛНИТЕЛЬНО,
answerParameter Container ДОПОЛНИТЕЛЬНО,
ret ReturnStatus ДОПОЛНИТЕЛЬНО
}

Если устройство DSRC не способно передать eriRequestPdu в ERT, в придорожный блок возвращается ответ, содержащий компонент ret типа ReturnStatus.

Примечание — Механизмы, используемые для передачи EriRequestPdu с устройства DSRC на устройство ERI, выходят за рамки настоящего стандарта. Предполагается, что появится общая бортовая платформа или сеть, которые могут быть использованы с этой целью. Тем временем изготовитель устройства DSRC, возможно, будет использовать различные средства для подключения своего устройства DSRC к встроенному считывателю/записывающему устройству блока ERI.

д) Внедрение положений ERI в определение уровня прикладного уровня DSRC

Информация о модуле ASN.1 приведена в А.2 приложения А.

6.3.4 Интерфейс удаленного доступа

Абонентский интерфейс ближнего действия должен иметь возможность обмена блоками протокола данных ERI в соответствии с 6.3.1.

Встроенное устройство связи, обеспечивающее удаленный доступ к ERT, должно предоставлять возможность передачи блоков данных протокола ERI, полученных от одноранговой сети (сотовой сети), в ERT и обратно.

Приложение А
(обязательное)

Модули ASN.1

А.1 Обзор

Это приложение содержит следующие модули ASN.1:
модуль транзакции;
уменьшенный модуль, для того чтобы показать, каким образом его можно использовать.
Информационный модуль ТС ElectronicRegistrationIdentificationVehicleDataModule приведен в ПНСТ 343–2018.

А.2 Модули

Примечание — Этот раздел можно в целом преобразовать в простой текст и затем скомпилировать, поэтому он не содержит заголовков и заголовков дополнительных предложений.

А.2.1 Модуль операций

ElectronicRegistrationIdentificationTransactionsModule
{iso (1) standard (0) iso24534 (24534) транзакции (2) версия (0)}
ОПРЕДЕЛЕНИЯ АВТОМАТИЧЕСКИЕ ТЕГИ :: = НАЧАТЬ

- Операции с электронной регистрацией идентификационных данных (ERI)
- ЭКСПОРТ все;

ИМПОРТ
RegistrationAuthority, VehicleId, AdditionalEriData, EntityId
FROM ElectronicRegistrationIdentificationVehicleDataModule;

```
EriPdu ::= CHOICE {
        requestPdu              EriRequestPdu,
        reponsePdu              EriResponsePdu
        }

EriRequestPdu ::= SEQUENCE {
        transactCode            TRANSACTION.&transactionCode ( {EriTransactions}),
        argument                TRANSACTION.&ArgumentType
                                ( {EriTransactions} {@.transactCode}) OPTIONAL
        }

EriResponsePdu ::= CHOICE {
        resultPdu               EriResultPdu,
        errorPdu                EriErrorPdu
        }

EriResultPdu ::= SEQUENCE {
        transactCode            TRANSACTION.&transactionCode ( {EriTransactions}),
        result                  TRANSACTION.&ResultType ( {EriTransactions}
        }                       {@.transactCode})
EriErrorPdu ::= SEQUENCE {
        transactCode            TRANSACTION.&transactionCode ( {EriTransactions}),
        error                   TRANSACTION.&ErrorCodes ( {EriTransactions}
        }                       {@.transactCode})
```

A.2.2 Операции

```
TRANSACTION ::= CLASS {
        &ArgumentType                   ,
        &ResultType                     ,
        &ErrorCodes                     ErrorCode OPTIONAL,
        &transactionCode                INTEGER UNIQUE
        }

WITH SYNTAX {
        ARGUMENT                        &ArgumentType
        RESULT                          &ResultType
        [ERRORS                         &ErrorCodes]
        CODE                            &transactionCode
        }

EriTransactions TRANSACTION ::= {
        getEriData                      |
getAuthenticatedEriData |
        setEriData      |   getCiphertextHistoricEriData    |   getCleartextHistoricEriData    |
        getPublicCertificateVerificationKeyId | getPublicEnciphermentKeyErt |
        commissionErt | withdrawCommissioning |
        getCiphertextHistoricComData | getCleartextHistoricComData |
        updateAccessControlList     |   getCiphertextAccessControlListEntry                |
getCleartextAccessControlListEntry |        getErtCapabilities
        }
```

A.2.3 Получить данные ERI

```
getEriData TRANSACTION ::= {
        ARGUMENT                        GetEriDataArgument
        RESULT                          GetEriDataResult
        ERRORS                          {notCustomized}
        CODE                            1
        }

GetEriDataArgument ::= SEQUENCE {
        onBehalfOf                      EntityId OPTIONAL,
        challenge                       Challenge,
        includeAdditionalData           BOOLEAN
        }

GetEriDataResult ::= SEQUENCE {
        registrationAuthority           RegistrationAuthority OPTIONAL,
        eriResultData                   SECURED {CleartextEriData}
        }
```

А.2.4 Аутентификация данных ERI

```
getAuthenticatedEriData TRANSACTION ::= {
        ARGUMENT                        GetAuthenticatedEriDataArgument
        RESULT                          GetAuthenticatedEriDataResult
        ERRORS                          {notCustomized}
        CODE                            2
        }

GetAuthenticatedEriDataArgument ::= SEQUENCE {
        ertHolderCredentials            ErtHolderCredentials,
        challenge                       Challenge,
        includeAdditionalData           BOOLEAN
        }

GetAuthenticatedEriDataResult ::= SEQUENCE {
        registrationAuthority   EntityId OPTIONAL,
        authenticateResultData          CLEAR-SECURED {CleartextEriData}
        }
```

А.2.5 Данные ERI и флаги безопасности ERT

```
 CleartextEriData ::= SEQUENCE {
        eriDataOrId                     EriDataOrId,
        ertSecurityStatus               ErtSecurityFlags OPTIONAL
        }

EriDataOrId ::= CHOICE {
        vehicleId                       VehicleId,
        unsignedDatedEriData            DATED {EriData},
        datedAndSignedEriData           SIGNED {DATED {EriData}, PrivateSignatureKey} -- BOE
        }                               signed

EriData ::= SEQUENCE {
        id                              VehicleId,
        additionalEriData               OCTET STRING (CONTAINING AdditionalEriData)
                                        OPTIONAL
        }
ErtSecurityFlags ::= BIT STRING {
        flagsHaveBeenResetted (0),   notCommissioned                (1),
        lowSupplyVoltageIndication (2),      highSupplyVoltageIndication
(3),    lowClockFrequencyIndication (4),    highClockFrequencyIndication
(5),    lowTemperatureIndication (6),   highTemperatureIndication (7)
        } (SIZE (0..16))                -- bit 8 .. 15 reserved for future use
```

A.2.6 Установка данных ERI

```
setEriData TRANSACTION ::= {
        ARGUMENT                    SetEriDataArgument
        RESULT                      NULL
        ERRORS                      {SetEriDataErrors}
        CODE                        3
        }

SetEriDataArgument ::= CHOICE {
        clearTextArgument           ClearTextSetEriDataArgument,
        encryptedArgument           ENCRYPTED
                                    {ClearTextSetEriDataArgument}
        }

ClearTextSetEriDataArgument ::= CHOICE {
        authenticatedEriData        BOE-AUTHENTICATED        {DATED
                                    {EriData}},
        notAuthenticatedEriData     DATED {EriData}
        }

SetEriDataErrors ErrorCode ::= {
        illegalArgument |           illegalVehicleId |
        illegalCertificate |        illegalSignature |
        illegalDate |               notCommissioned |
        resourceLimitExceeded |     otherError
        }
```

A.2.7 Получение исторических данных ERI

```
getCiphertextHistoricEriData TRANSACTION ::= {
        ARGUMENT                    GetCiphertextHistoricEriDataArgument
        RESULT                      SECURED {HistoricEriData}
        ERRORS                      {notCustomized}
        CODE                        4
       }

GetCiphertextHistoricEriDataArgument ::= SEQUENCE {
        onBehalfOf          EntityId OPTIONAL,
        challenge           Challenge,
        number              INTEGER (1..int4)
       }

getCleartextHistoricEriData TRANSACTION ::= {
        ARGUMENT                    GetCleartextHistoricEriDataArgument
        RESULT                      CLEAR-SECURED {HistoricEriData}
        ERRORS                      {notCustomized}
```

```
        CODE                                5
        }

GetCleartextHistoricEriDataArgument ::= SEQUENCE {
        credentials                         ErtHolderCredentials,
        challenge                           Challenge,
        number                              INTEGER (1..int4)
        }

HistoricEriData ::= SEQUENCE {
        number                              INTEGER (1..int4),
        outOf                               INTEGER (1..int4),
        historicRecord                      ClearTextSetEriDataArgument
        }
```

A.2.8 Получение общедоступного ключа проверки

```
getPublicCertificateVerificationKeyId TRANSACTION ::= {
        ARGUMENT                            NULL
        RESULT                              KeyId
        CODE                                6
      }

getPublicEnciphermentKeyErt TRANSACTION ::= {
        ARGUMENT                            BOE-AUTHENTICATED
                                            {VehicleId}
        RESULT                              PublicEnciphermentKey
        ERRORS                              {GetPublicEnciphermentKeyErrors}
        CODE                                7
      }

GetPublicEnciphermentKeyErrors ErrorCode ::= {
        illegalArgument |                   illegalVehicleId |
        illegalCertificate |                illegalSignature |
        illegalEntity |                     noDeciphermentCapability |
        otherError                          }
```

A.2.9 Введение в эксплуатацию

```
commissionErt TRANSACTION ::= {
        ARGUMENT                            CommissionErtArgument
        RESULT                              NULL
        ERRORS                              {CommissionErtErrors}
        CODE                                8
        }
```

```
CommissionErtArgument ::= CHOICE {
        authenticatedData                   BOE-AUTHENTICATED {DATED
                                            {CommissioningData}},
        notAuthenticatedData                DATED {CommissioningData}
        }
CommissioningData ::= SEQUENCE {
        vehicleId                           VehicleId,
        registrationAuthority               EntityId,
        resetSecurityFlags                  BOOLEAN,
        enciphermentKeyId                   KeyId OPTIONAL,
        publicEnciphermentKeyAuthority      PublicEnciphermentKey OPTIONAL,
        publicVerificationKeyCertificate    ErtCertificate OPTIONAL,
        privateData     ENCRYPTED {PrivateCommissioningData} OPTIONAL
        }
PrivateCommissioningData ::= SEQUENCE {
        privateSignatureKeyErt              PrivateSignatureKey
                                            OPTIONAL,
        pin                                 PIN OPTIONAL
        }

CommissionErtErrors ErrorCode ::= {
        illegalArgument |                   illegalVehicleId |
        illegalCertificate |                illegalSignature |
        illegalEntity |                     illegalDate |
        notCustomized |                     resourceLimitExceeded |
        noEnciphermentCapability |          secretKeyEncryptionAlgorithmNotSupported |
        publicKeyEncryptionAlgorithmNotSupported |
        noSigningCapability |       hashingAlgorithmNotSupported |
        signingAlgorithmNotSupported |
        otherError      }
```

A.2.10 Ввод в эксплуатацию

```
withdrawCommissioning TRANSACTION ::= {
        -- this transaction also removes all public encipherment keys
        ARGUMENT                            WithdrawCommissioningArgument
        RESULT                              SECURED
                                            {WithdrawCommissioningResultData}
        ERRORS                              {WithdrawCommissioningErrors}
        CODE                                withdrawCommissioningCode
        }

withdrawCommissioningCode INTEGER ::= 9

WithdrawCommissioningArgument ::= CHOICE {
authenticatedData BOE-AUTHENTICATED {VehicleId}, notAuthenticatedData VehicleId
        }
```

```
WithdrawCommissioningResultData ::= [APPLICATION withdrawCommissioningCode ] SEQUENCE {
        withdrawn          WithdrawCommissioningArgument, historicComData
        HistoricComData
        }

WithdrawCommissioningErrors ErrorCode ::= {
        illegalArgument |                   illegalVehicleId |
        illegalCertificate |                illegalSignature |
        illegalEntity |                     illegalDate |
        notCustomized |                     notCommissioned |
        otherError                          }
```

A.2.11 Получение исторических данных ввода в эксплуатацию

```
getCiphertextHistoricComData TRANSACTION ::= {
        ARGUMENT                            GetCiphertextHistoricComDataArgument
        RESULT                              SECURED {HistoricComData}
        ERRORS                              {notCommissioned}
        CODE                                9
      }

GetCiphertextHistoricComDataArgument ::= SEQUENCE {
        onBehalfOf                          EntityId
                                            OPTIONAL,
        challenge                           Challenge,
        number                              INTEGER (1..int4)
      }

getCleartextHistoricComData TRANSACTION ::= {
        ARGUMENT                            GetCleartextHistoricComDataArgument
        RESULT                              CLEAR-SECURED {HistoricComData}
        ERRORS                              {notCommissioned}
        CODE                                10
      }

GetCleartextHistoricComDataArgument ::= SEQUENCE {
        credentials                         ErtHolderCredentials,
        challenge                           Challenge,
        number                              INTEGER (1..int4)
      }

HistoricComData ::= SEQUENCE {
        number                              INTEGER (1..int4),
        outOf                               INTEGER (1..int4),
        historicRecord                      CommissionErtArgument
      }
```

A.2.12 Обновление списка управления доступом

```
updateAccessControlList TRANSACTION ::= {
        ARGUMENT                        UpdateAccessControlListArgument
        RESULT                          NULL
        ERRORS                          {UpdateAccessControlListErrors}
        CODE                            11
      }

UpdateAccessControlListArgument ::= CHOICE {
        authorityUpdate   SIGNED {AccessControlListUpdate, PrivateSignatureKey},
        holderUpdate      HolderAccessControlListUpdate
        }
HolderAccessControlListUpdate ::= SEQUENCE {
        credentials       ErtHolderCredentials,
        entry     AccessControlListUpdate
        }

AccessControlListUpdate ::= SEQUENCE {
        mode                            ENUMERATED {deleteAllAndAdd (0), add (1), delete
                                        (2)}
                                        DEFAULT add,
        entry                           AccessControlEntry OPTIONAL
         }

        AccessControlEntry ::= SEQUENCE {
        id                              EntityId,
        name                            Text OPTIONAL,
        enciphermentKeyId               KeyId OPTIONAL,
        publicEnciphermentKeyReader     PublicEnciphermentKey OPTIONAL
        }

UpdateAccessControlListErrors ErrorCode ::= {
        illegalArgument |  illegalVehicleId |
        illegalSignature |  illegalHolderAccess |
        illegalDate |       noEntry |
        resourceLimitExceeded |    noEnciphermentCapability |
        otherError         }
```

A.2.13 Получение списка управления доступом

```
getCiphertextAccessControlListEntry TRANSACTION ::= {
        ARGUMENT                        GetCiphertextAccessControlListEntryArgument
        RESULT                          SECURED {AuthorityAccessControlListEntry}
        CODE                            12
        }

GetCiphertextAccessControlListEntryArgument ::= SEQUENCE {
        challenge         Challenge,
        number            INTEGER (1..int4)
        }
```

```
getCleartextAccessControlListEntry TRANSACTION ::= {
        ARGUMENT                        GetCleartextAccessControlListEntryArgument
        RESULT                          CLEAR-SECURED {AccessControlListEntry}
        ERRORS                          {GetCleartextAccessControlListEntryErrors}
        CODE                            13
        }

GetCleartextAccessControlListEntryArgument ::= SEQUENCE {
        credentials                     ErtHolderCredentials,
        challenge                       Challenge,
        number                          INTEGER (1..int4)
        }
AuthorityAccessControlListEntry ::= AccessControlListEntry (WITH COMPONENTS {..., holderEntry
ABSENT} ) AccessControlListEntry ::= SEQUENCE {
        number                          INTEGER (0..int4),
        outOf                           INTEGER (0..int4),
        authorityEntry                  AccessControlEntry
                                        OPTIONAL,
        holderEntry                     AccessControlEntry
                                        OPTIONAL
        }

GetCleartextAccessControlListEntryErrors ErrorCode ::= {
        illegalArgument |                       illegalVehicleId |
        illegalHolderAccess |                   otherError
        }
```

A.2.14 Получение возможностей ERT

```
getErtCapabilities TRANSACTION ::= {
        ARGUMENT                        NULL
        RESULT                          ErtCapabilities
        CODE                            15
        }

ErtCapabilities ::= SEQUENCE {
        supportedTransactions           SEQUENCE OF INTEGER,
        hashingAlgorithms               SEQUENCE OF HashingAlgorithm OPTIONAL,
        signingAlgorithms               SEQUENCE OF PublicKeyAlgorithm OPTIONAL,
        signatureVerificationAlgorithms SEQUENCE OF PublicKeyAlgorithm OPTIONAL,
        secretKeyEncryptionAlgorithms   SEQUENCE OF SecretKeyAlgorithm OPTIONAL,
        publicKeyEncryptionAlgorithms   SEQUENCE OF PublicKeyAlgorithm OPTIONAL,
        publicKeyDecryptionAlgorithms   SEQUENCE OF PublicKeyAlgorithm OPTIONAL,
        maxBitsPublicKeys               INTEGER (0..int4),
        maxOctetsPin                    INTEGER (0..int4),
        maxOctetsSetArgument            INTEGER (0..int4),
        maxNumberSetArguments           INTEGER (1..int4),
```

```
        maxNumberComArguments           INTEGER (1..int4),
        maxSizeAccessControlList        INTEGER (0..int4),
        maxNumberAuthorities            INTEGER (0..int4),
        maxNumberAddServProviders       INTEGER (0..int4),
        ertSecurityIndicationSupport    ErtSecurityFlags,
        maxInteger                      INTEGER (1..int4),
        maxStringSize                   INTEGER (1.. int4),
        ...
        }
```

A.2.15 Маркировка и аутентификация ERT

```
SECURED {ToBeSecured} ::= CHOICE {
        authenticatedAndEncrypted       ENCRYPTED {ERT-AUTHENTICATED {TAGGED
{ToBeSecured}}}, authenticated      ERT-AUTHENTICATED {TAGGED {ToBeSecured}},
        encrypted           ENCRYPTED {TAGGED {ToBeSecured}},

        cleartext           TAGGED {ToBeSecured}
        }

CLEAR-SECURED {ToBeSecured} ::= SECURED {ToBeSecured} (WITH COMPONENTS
        {authenticatedAndEncrypted ABSENT, encrypted ABSENT} )

TAGGED {ToBeTagged} ::= SEQUENCE {
        ertNumber                       ErtNumber,
        challenge                       Challenge OPTIONAL,
        sequenceNumber                  INTEGER         (1..int4)
                                        OPTIONAL,
        tobeTagged                      ToBeTagged
        }

ERT-AUTHENTICATED {ToBeErtAuthenticated} ::= SEQUENCE {
        ertSigned           SIGNED {ToBeErtAuthenticated, PrivateSignatureKey},
        publicVerificationKeyCertificate ErtCertificate
        }

ErtCertificate ::= SIGNED {ErtCertificationData, PrivateSignatureKey}

ErtCertificationData ::= SEQUENCE {
        vehicleId                       VehicleId,
        publicVerificationKey           PublicVerificationKey,
        signatoryId                     EntityId,
        date                            DATE
        }
```

А.2.16 Даты и аутентификация BOE

```
DATED {ToBeDated} ::= SEQUENCE {
        date      DATE,
validThru DATE OPTIONAL, issuer          EntityId,
        toBeDated       ToBeDated
        }

BOE-AUTHENTICATED {ToBeBoeAuthenticated} ::= SEQUENCE {
signedParameter SIGNED {ToBeBoeAuthenticated, PrivateSignatureKey}, certificates SEQUENCE
SIZE(1..2) OF BoeCertificate
        }

BoeCertificate ::= SIGNED {BoeCertificationData, PrivateSignatureKey}
BoeCertificationData ::= SEQUENCE {
entityId EntityId, entityRole EntityRole, entityName Text
OPTIONAL, publicKey Key,
        signatoryId       EntityId, signatoryName   Text
OPTIONAL,       signatoryRole   EntityRole,   date   DATE,
        validThru         DATE
        }

EntityRole::= ENUMERATED {
        topLevelCertificationAuthority (0), intermediateCertificationAuthority (1),
        manufacturer (2),                   registrationAuthority
                                            (3),
        authority (4),                      serviceProvider (5),
        eriHolder (6)                       }
```

А.2.17 Подписание

```
SIGNED {ToBeSigned, PrivateSignatureKey} ::= SEQUENCE {
        toBeSigned                          ToBeSigned,
        hashingAlgorithm      HashingAlgorithm DEFAULT sha1,
        signatureAlgorithm                  PublicKeyAlgorithm DEFAULT ellipticCurve,
        signature             SIGNATURE {ToBeSigned, HashingAlgorithm,
                                            PublicKeyAlgorithm , PrivateSignatureKey}
        }

        SIGNATURE {ToBeSigned, HashingAlgorithm, SignatureAlgorithm, PrivateSignatureKey} ::=
        BIT STRING (CONSTRAINED BY
{HashingAlgorithm, -- and -- SignatureAlgorithm, -- with -- PrivateSignatureKey, -- applied to --
ToBeSigned} )

Challenge ::= INTEGER (1..int4)
```

А.2.18 Шифрование

```
ENCRYPTED {ToBeErtEncrypted} ::= SEQUENCE {
    enciphermentKeyId       KeyId,
    publicKeyEncryptionAlgorithm    PublicKeyAlgorithm DEFAULT ellipticCurve,
    secretKeyEncryptionAlgorithm    SecretKeyAlgorithm DEFAULT aes,
```

```
        encryptedKey      KEY-ENCRYPTION {SecretTransactionKey,
        PublicKeyAlgorithm, PublicEnciphermentKey},        ciphertext        CIPHER-TEXT
{ToBeErtEncrypted,                SecretKeyAlgorithm, SecretTransactionKey}
        }

KEY-ENCRYPTION {KeyToBeEncrypted, PublicKeyAlgorithm, PublicEnciphermentKey} ::=
        BIT STRING ( CONSTRAINED BY {
        PublicKeyAlgorithm, -- with -- PublicEnciphermentKey, -- applied to -- KeyToBeEncrypted
        -- or random bit string with same length if no public key is available --} )

CIPHER-TEXT {ToBeEncrypted, SecretKeyAlgorithm, SecretKey} ::=
        BIT STRING ( CONSTRAINED BY {
        SecretKeyAlgorithm, -- with -- SecretKey, -- applied to -- ToBeEncrypted
        -- or random bit string with same length if no public key is available --} )
```

A.2.19 Алгоритмы шифрования

```
HashingAlgorithm ::= ENUMERATED {
    sha1,
    ... }

PublicKeyAlgorithm ::= ENUMERATED {
        ellipticCurve,
        ... }

SecretKeyAlgorithm ::= ENUMERATED {
        aes,
        ... }
```

A.2.20 Учетные данные и ключи

```
ErtHolderCredentials ::= SEQUENCE {
        vehicleId        VehicleId,
        pin        PIN
        }

SecretTransactionKey    ::=        Key
PublicEnciphermentKey        ::=   Key
PrivateDeciphermentKey    ::=     Key
PrivateSignatureKey       ::=     Key
PublicVerificationKey ::= Key
Key                ::= BIT STRING

PIN                ::= NumericString (SIZE(4))

KeyId              ::= INTEGER (0..65535)
```

A.2.21 Идентификатор ERT

```
ErtNumber ::= INTEGER
```

A.2.22 Текст

```
Text ::= UTF8String (SIZE (1..256))

-- int4

int4 INTEGER ::= 4294967295
```

А.2.23 Коды ошибок

```
ErrorCode ::= ENUMERATED {
        illegalArgument,                    illegalVehicleId,
        illegalCertificate,                 illegalSignature,
        illegalEntity,                      illegalHolderAccess,
        illegalDate,                        notCustomized,
        notCommissioned,                    noEntry,
        resourceLimitExceeded,
-- encipherment support errors
        noEnciphermentCapability,
        noDeciphermentCapability,
        secretKeyEncryptionAlgorithmNotSupported,
        publicKeyEncryptionAlgorithmNotSupported,
-- authentication support errors
        noSigningCapability,
        noSignatureVerificationCapability,
        hashingAlgorithmNotSupported,
        signingAlgorithmNotSupported,

        otherError,
        ... }
END
```

Приложение Б
(обязательное)

Форма протоколов PICS

Б.1 Введение

В настоящем приложении содержатся проформы формы соответствия протокола (PICS), которые будут использоваться для ERT и считывателей и писателей ERI.

Б.2 Поддержка транзакций

Настоящее приложение применяется как к ERT, так и к читателям или писателям ERI.

GetEriData

Дополнительная поддержка данных Eri	Да/нет
Время ответа ERT для идентификатора TC, мс	
Время ответа ERT для записи данных ERi максимальной длины, мс	

GetAuthenticatedEriData

Поддерживает	Да/нет
Дополнительная поддержка данных Eri	Да/нет

SetEriData

Поддерживает	Да/нет
Дополнительная поддержка данных Eri	Да/нет

GetCiphertextHistoricEriData

Поддерживает	Да/нет

GetCleartextHistoricEriData

Поддерживает	Да/нет

GetPublicCertificateVerificationKeyId

Поддерживает	Да/нет

GetPublicEnciphermentKeyErt

Поддерживает	Да/нет

CommissionErt

Услуги конфиденциальности	Да/нет
Поддержка PIN-кода держателя ERT	Да/нет

WithdrawCommission

Поддерживает	Да/нет

GetCiphertextHistoricComData

Поддерживает	Да/нет

GetCleartextHistoricComData

Поддерживает	Да/нет

UpdateAccessControlList

Поддерживает	Да/нет
Поддержка ввода в эксплуатацию	Да/нет
Дополнительная поддержка авторизации поставщика услуг	Да/нет

Примечание — Эта транзакция имеет смысл только тогда, когда поддерживается конфиденциальность.

GetCiphertextAccessControlListEntry

Поддерживает	Да/нет

GetCleartextAccessControlListEntry

Поддерживает	Да/нет

ErtCapabilities

Поддерживает	Да/нет

Возможности безопасности
Услуги безопасности
Этот пункт распространяется как на ERT, так и на считывателей или писание ERI.

Описание	Значение(я)
Подписание поддержки (ERT: чтение транзакций, сценаристы: запись транзакций)	Да/нет
Поддержка проверки подписи (ERT: транзакции записи, считыватели: чтение транзакций)	Да/нет
Поддержка шифрования (ERT: чтение транзакций, сценаристы: запись транзакций)	Да/нет
Поддержка дешифрования (ERT: транзакции записи, считыватели: чтение транзакций)	Да/нет
Поддерживаемые алгоритмы хеширования (для подписания)	—
Поддерживаемые алгоритмы секретного ключа	—
Поддерживаемые алгоритмы секретного ключа	—

Поддержка мониторинга состояния безопасности
Этот пункт относится только к ERT.

Описание	Значение(я)
Обнаружение низкого напряжения	Да/нет
Обнаружение высокого напряжения	Да/нет
Низкочастотное обнаружение частоты	Да/нет

Описание	Значение(я)
Обнаружение высокой частоты	Да/нет
Обнаружение низкой температуры	Да/нет
Обнаружение высокой температуры	Да/нет

Б.3 Емкость хранилища ERT

Этот пункт применим исключительно к ERT.

Б.3.1 Емкость хранения данных ERI

Описание	Максимальные значение или диапазон
Максимальный размер записи данных ERI	
Объем памяти для записей аргументов SetEriData (число)	

Б.3.2 Емкость хранения данных комиссии

Описание	Максимальные значение или диапазон
Максимальный орган регистрации	
Максимальный ключ частной секретной подписи ERT	
Максимальный удлинитель PIN ERT	
Емкость записей CommisionErt (число)	

Б.3.3 Объем памяти для контроля доступа

Описание	Максимальные значение или диапазон
Максимальные длинные ключи общедоступного шифрования органов власти и дополнительных поставщиков услуг	
Максимальное количество полномочий, которые могут быть разрешены	
Максимальное количество дополнительных поставщиков услуг, которые могут быть разрешены	

Б.3.4 Общие значения

Описание	Максимальные значение или диапазон
Целые числа (минимальное и максимальное)	
Строки (максимальный размер)	

Приложение В
(справочное)

Эксплуатационные сценарии

В.1 Сценарий настройки и ввода в эксплуатацию

В.1.1 Сценарий

Настройка и ввод в эксплуатацию ERT соответствует приведенному на рисунке В.1.

Действующее лицо. Условие проведения действия	Действия ограничения/применения
Изготовление устройств: Действующее лицо: производитель ERT Условия: - когда ERT изготовлена	Добавленные: - общедоступный контрольный ключ и ключевой идентификатор центра сертификации высшего уровня; - асимметричная пара ключей для ввода в эксплуатацию шифрования транзакций Ограничение/примечание: - внешний частный ключ дешифрования не будет доступен
Индивидуальная настройка (для конкретного ТС) Действующие лица: производитель или регистрирующий орган Условия: - когда ТС сконструировано или - когда установлен новый компонент	Добавленные: - данные ТС и дополнительные данные, при их наличии Ограничения/примечания: Это действие может быть выполнено только один раз. Одно или два свидетельства должны отозвать действующее лицо. Один сертификат выдается центром сертификации высшего уровня. Второй, если имеется, выдается промежуточным центром сертификации, для которого выдан первый сертификат
Введение в эксплуатацию Действующее лицо: регистрирующий орган Условия: - когда данные о безопасности добавлены или обновлены	Добавленные или измененные: - идентификатор регистрирующего органа; - ключ частной подписи и идентификатор ключа для данных ERI (если требуется аутентификация); - если общедоступный ключ шифрования и идентификатор ключа для защиты конфиденциальности (при необходимости считыватели должны быть введены в экплуатацию с соответствующим секретным ключом дешифрования); - PIN-код, который должен быть выдан держателю ERT для доступа к полному доступу (если требуется конфиденциальность) Ограничения/примечания: - ТС в данных ERI должно быть таким же, как и для настройки; - для проверки подлинности

Рисунок В.1 — Настройка и ввод в эксплуатацию ERT

В.1.2 Изготовление устройства

Для проверки сертификатов, выданных центром сертификации высшего уровня, требуется общедоступный контрольный ключ.

В.1.3 Настройка

Контроль доступа выполняется с соблюдением такого требования, при котором заказчик подписал идентификатор ТС и доказал свои права доступа путем предоставления сертификата, выпущенного центром сертификации высшего уровня. Заказчику требуется для предоставления только один сертификат, если он получен непосредственно от центра сертификации высшего уровня, и два, если они получены от промежуточного центра сертификации. В последнем случае один из сертификатов должен быть сертификатом, полученным центром сертификации высшего уровня для ключа общедоступного шифрования промежуточного центра сертификации.

Когда ERT не поддерживает проверку подписи, сертификат(ы) не проверяет(ют)ся.

Примечание — Так как данные ERI не содержат конфиденциальных элементов, таких как закрытый ключ или личные данные, отсутствует необходимость шифровать данные ERI при настройке устройства ERI.

B.1.4 О вводе в эксплуатацию

Информация для контроля доступа приведена в B.1.3.

Когда ERT не поддерживает шифрование, общедоступные ключи шифрования не должны заменяться. Когда ERT не поддерживает проверку подписи, сертификаты и секретный ключ подписи для подписи данных ERI ERT не должны подлежать обмену.

Если ERT не поддерживает ни аутентификацию, ни конфиденциальность, не должны применяться меры безопасности к транзакции ввода в эксплуатацию; данные затем могут быть непосредственно записаны в ERT.

Примечание — Любой регистрирующий орган может (повторно) поручить любой компонент ERI, содержащий данные ERI. Это соответствует текущей практике, согласно которой любой регистрирующий орган может предоставить новую номерную табличку для ТС (независимо от того, соответствует ли это практике международных соглашений, так как выходит за рамки настоящего стандарта).

B.1.5 Схема перехода состояния ERT

На рисунке B.2 представлена диаграмма состояния ERT в отношении настройки и ввода в эксплуатацию.

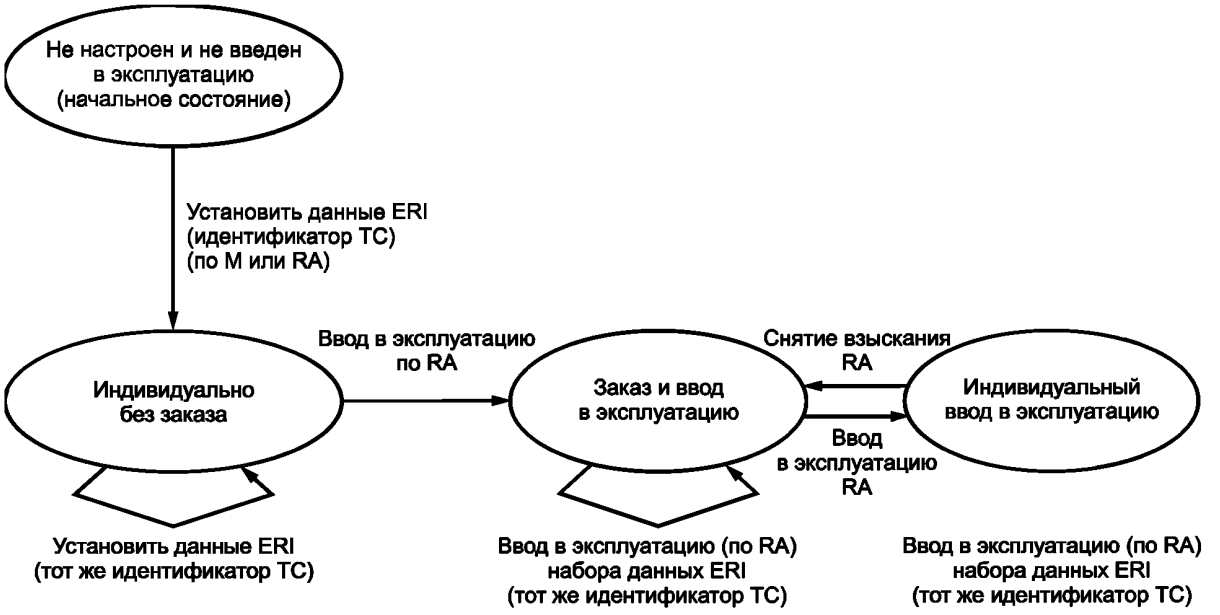

Рисунок B.2 — Диаграмма состояния ERT

Примечание — Переходы состояния вызываются производителями (M) или регистрирующими органами (RA).

B.1.6 Сценарии ERI с различными уровнями безопасности

B.1.6.1 Контекст

Различные регистрирующие органы могут требовать разного уровня безопасности. В соответствии с этими требованиями должна быть современная служба аутентификации, конфиденциальности и контроля доступа; однако эти условия могут быть не соблюдены.

В настоящем пункте приведены сценарии для ERI с различными уровнями безопасности. Эти сценарии совместимы с повышенным уровнем и в некоторой степени с пониженными уровнями безопасности. ТС с устройством ERI с меньшей степенью защиты можно идентифицировать с помощью оборудования, предназначенного для устройств с большей защитой.

Для сценариев ERI приведены уровни безопасности ERT, поддерживающие:

аутентификацию и конфиденциальность;

аутентификацию, но не конфиденциальность;

конфиденциальность, но не аутентификацию, а также ERT, не поддерживающие ни аутентификацию, ни конфиденциальность (т. е. только для ТС).

Для упрощения приведенных ниже сценариев компонент «Дополнительные данные о ТС» не отображается (и считается ложным).

В.1.6.2 ERI с ERT, поддерживающей как аутентификацию, так и конфиденциальность

Если компонент ERI, который содержит данные ERI, поддерживает как аутентификацию, так и конфиденциальность, ERI работает согласно схеме, приведенной на рисунке В.3.



Рисунок В.3 — ERI из ERT, которая поддерживает как конфиденциальность, так и аутентификацию

Примечания

1 Если идентификатор полномочий опущен или отсутствует в списке управления доступом, идентификатор по умолчанию является регистрирующим органом, который поручил ERT.

2 В критичных по времени ситуациях этот секретный ключ транзакции уже может быть зашифрован до того, как будет вызвана транзакция, для которой она необходима.

В.1.6.3 ERI с поддержкой ERT, но не конфиденциальность

Если компонент ERI, который содержит данные ERI, поддерживает аутентификацию, но не конфиденциальность, ERI работает согласно схеме, приведенной на рисунке В.4.

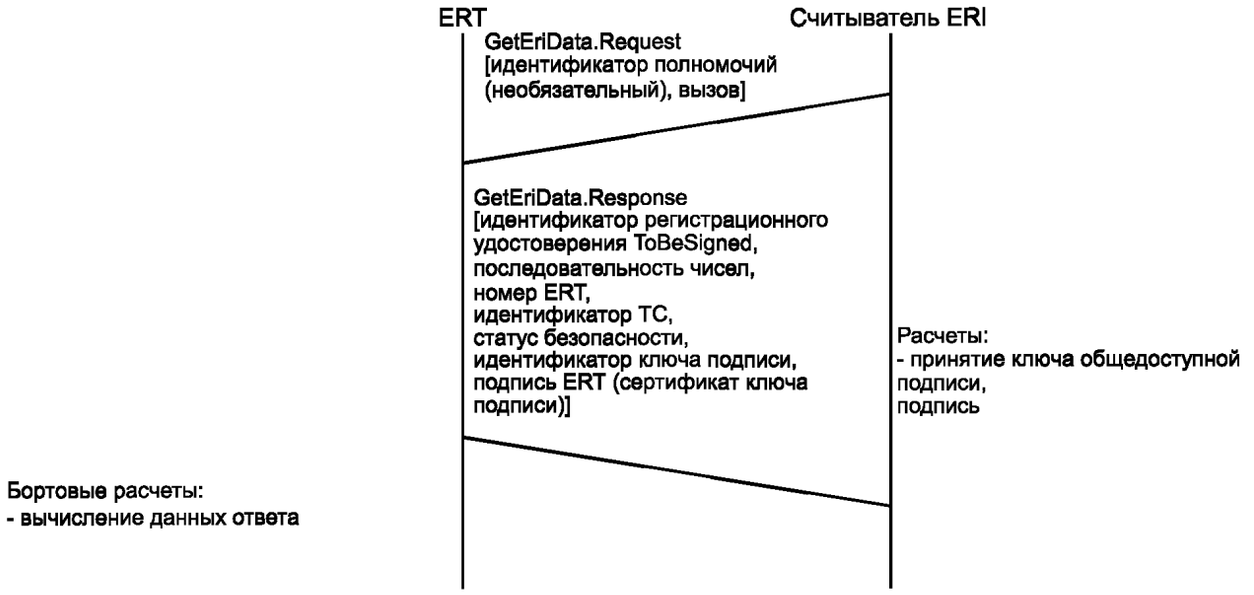

Рисунок В.4 — ERI из ERT, которая поддерживает исключительно аутентификацию

Примечания

1 Идентификатор полномочий в этом случае не используется ERT.

2 Номер последовательности не нужен, но включен в подписанную часть ответа.

В.1.6.4 ERI с ERT, поддерживающей конфиденциальность, но не аутентификацию

Если компонент ERI, который содержит данные ERI, поддерживает конфиденциальность, но не аутентификацию, ERI работает согласно схеме, приведенной на рисунке В.5.

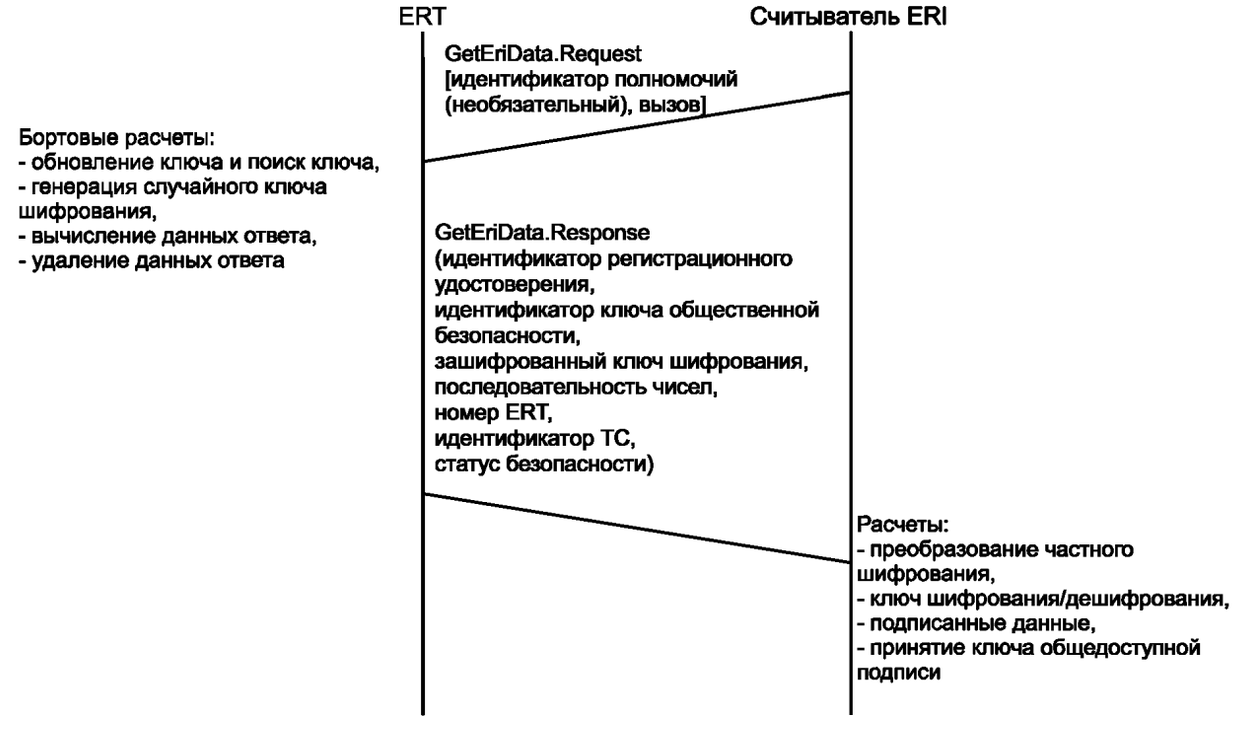

Рисунок В.5 — ERI из ERT, которая поддерживает исключительно конфиденциальность

Примечания

1 Если идентификатор полномочий опущен или отсутствует в списке управления доступом, идентификатор по умолчанию является регистрирующим органом, который поручил ERT.

2 Ситуация, при которой поддерживается конфиденциальность, но не аутентификация, маловероятна. Тем не менее она включена, чтобы показать совместимость ERI с различными уровнями безопасности.

3 Если аутентификация абсолютно не поддерживается, проблема может быть исключена из ответа.

B.1.6.5 ERI с ERT, не поддерживающей ни аутентификацию, ни конфиденциальность

Если компонент ERI, который содержит данные ERI, не поддерживает ни аутентификацию, ни конфиденциальность, ERI работает согласно схеме, приведенной на рисунке B.6.

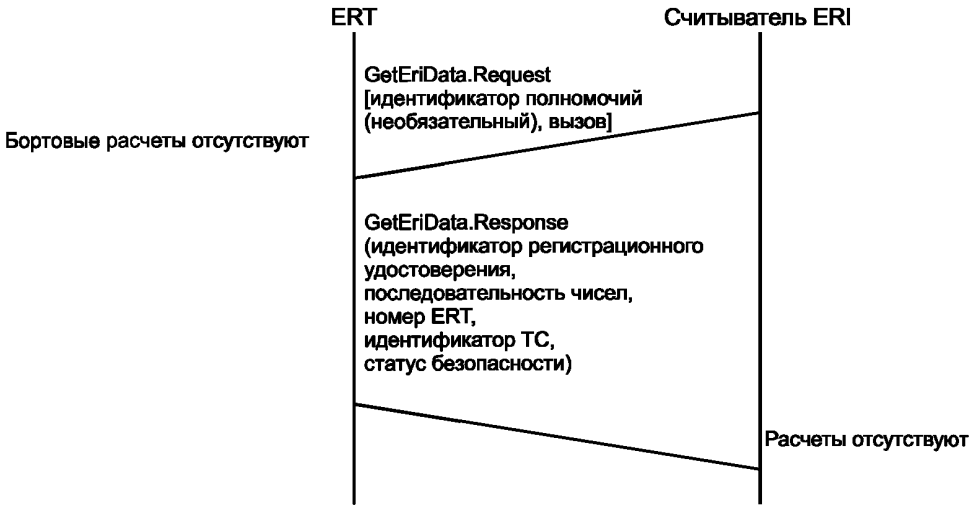

Рисунок B.6 — ERI из ERT при отсутствии служб безопасности

Примечания

1 Так как ERT не требует вычисления, это может быть довольно простое устройство.

2 Очень простой вариант ERT может даже опустить вызов из ответа.

3 Если вызов не используется в ответе, единственные накладные расходы для обеспечения совместимости с более сложными устройствами ERI находятся в параметрах «Идентификатор» (необязательный) и «Вызов» запроса. Это только усложняет считыватель ERI в отличие от считывателя ERT.

B.1.6.6 Требования

В таблице B.1 приведены параметры ответа в запросе данных ERI, которые необходимы для поддержки конфиденциальности или аутентификации.

Таблица B.1 — Данные ответа ERI и службы безопасности

Параметр	Аутентификация	Конфиденциальность	Постоянство
Идентификатор регистрирующего органа	—	—	R
Идентификатор открытого ключа	—	R	—
Зашифрованный ключ шифрования	—	R	—
Последовательность чисел	—	R	—
Вызов	R	—	—
Номер ERT	—	—	R
Идентификатор ТС	—	—	R
Состояние безопасности	—	—	R

Окончание таблицы В.1

Параметр	Аутентификация	Конфиденциальность	Постоянство
Идентификатор ключа подписи	R	—	—
Подпись ERT	R	—	—
Сертификат ключа подписи	O	—	—
Шифрование (подписанных) данных	—	R	—
Примечание — R — требуется; O — обязательный.			

В.1.6.7 Обеспечение соблюдения правил и других услуг. Применение скорости движения траектории. Контекст

Дорожный оператор или полиция хотят обеспечить ограничение скорости по траектории дорожной сети. С этой целью в начале и в конце траектории устанавливают считыватели ERI, способные обнаруживать расстояние до ТС. Так как расстояние между этими точками известно, можно сделать вывод о том, что каждое ТС, для которого время между первым и вторым считывателями ниже определенного значения, имеет среднюю скорость выше допустимого предела.

В.1.6.8 Сценарий

Сценарий работает следующим образом:

- для каждого ТС, входящего в траекторию, регистрируются идентификатор ТС и время входа. После того как минимальное время для движения траектории плюс некоторое время для идентификации иностранных ТС истекает, эти данные отбрасываются;
- для ТС, зарегистрированного в другом регистрирующем органе, применяется одна из следующих возможностей:
- считыватель ERI уполномочен для идентификации ТС от этого регистрирующего органа, или
- регистрирующему органу, который зарегистрировал ТС, предлагается расшифровать данные, полученные от ТС, для получения идентификатора ТС.

Для каждого ТС, покидающего траекторию движения, регистрируются идентификатор ТС и время выхода, а входной регистр проверяется на входную регистрацию того же ТС в течение установленного времени. Если идентификатор ТС не найден, данные выхода будут отброшены; если обнаружен, процедура начинается с начисления штрафа владельцу или хранителю ТС.

Для ТС, зарегистрированного в другом регистрирующем органе, используется та же процедура, что и при входе в траекторию, она может быть применена на выходе.

В.1.6.9 Требования

ТС вблизи движущихся местных и иностранных ТС должно быть идентифицировано при прохождении.

Примечания

1 Так как не имеет значения, на какой полосе движения ТС входит или выходит из траектории, требуется только идентификация ближайшей точки. Даже идентификация ТС на соседней проезжей полосе не вызовет никаких проблем до тех пор, пока движение на смежной проезжей полосе не будет работать в одном направлении с более высоким пределом скорости. Так как правила ЕС требуют, чтобы принудительное исполнение было недискриминационным, на территории ЕС читатели ERI должны, по крайней мере, иметь возможность идентифицировать ТС из других государств — членов ЕС.

2 Идентификация иностранных ТС не является критичной по времени, если не требуется их задержание.

В.1.6.10 Контроль доступа

Контекст

Дорожный оператор или полиция хотят обеспечить соблюдение нарушений на полосе HOV или HGV с помощью автоматических средств.

Сценарий

Сценарий приведен на рисунке В.7:

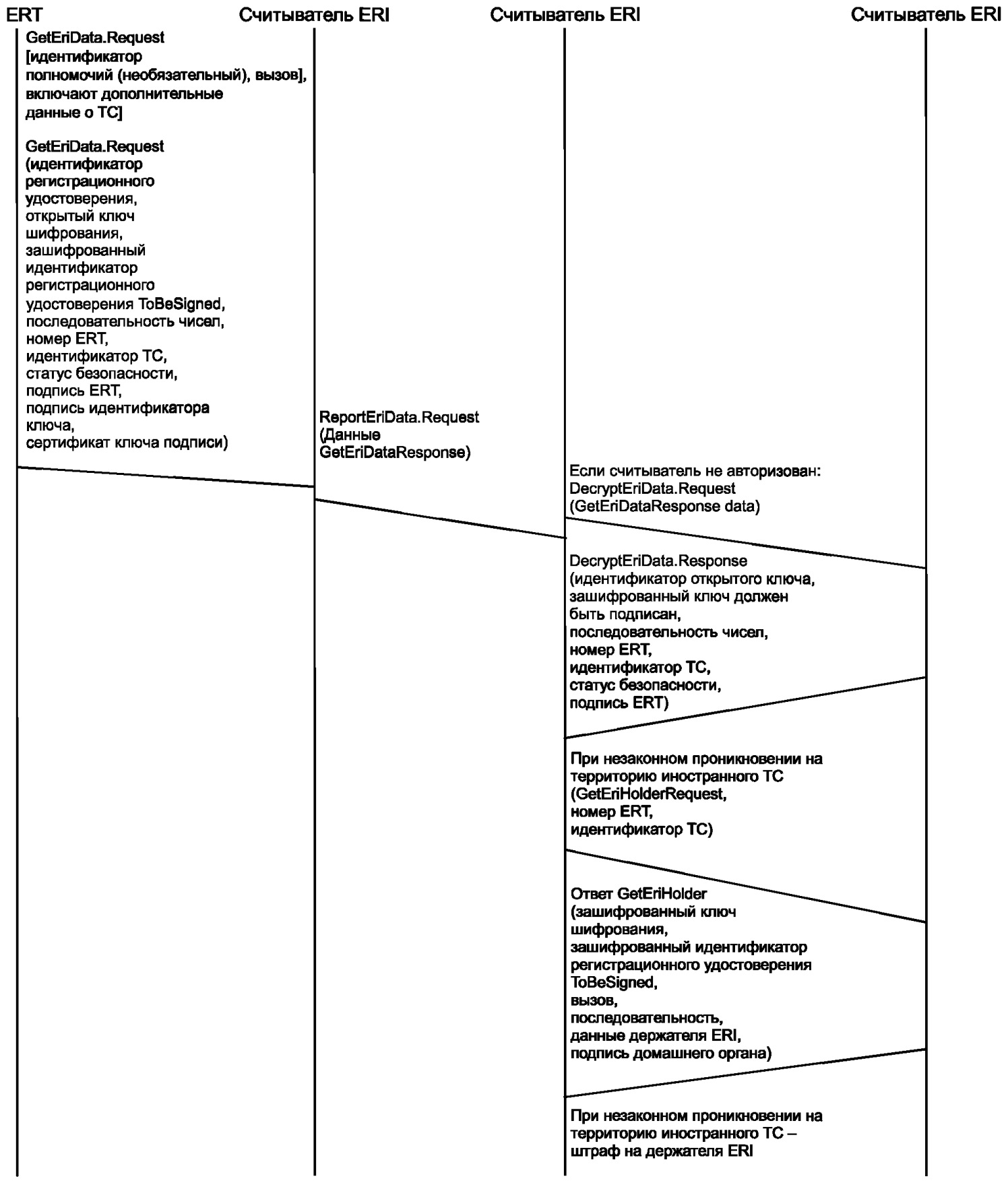

Рисунок В.7 — Принуждение к нарушению полосы HOV или HGV

GetEriData.Request

Операция получения данныхERI вызывается считывателем ERI, когда датчик обнаруживает новое ТС в полосе. Идентификатор полномочий идентифицирует орган, который поручил считыватель ERI.

GetEriData.Response

Результат шифруется с помощью ключа общедоступного шифрования органа, который заказал считыватель ERI, если он доступен в списке управления доступом к компоненту ERI, содержащему данные ERI. В противном случае он зашифрован с помощью ключа общедоступного шифрования регистрирующего органа, который поручил компонент ERI, содержащий данные ERI.

Данные ERI подписываются с ключом частной подписи компонента ERI, содержащего данные ERI.

ReportEriData

Если считыватель не может определить, разрешено ли ТС использовать полосу движения, он будет сообщать данные ERI ТС на локальное BOE.

Примечание — Считыватель может только определить, что ТС разрешено использовать в том случае, если данные ERI зашифрованы с помощью общедоступного ключа шифрования органа, который поручил читателю, и если дополнительные данные ТС содержат необходимую информацию (например, факт того, что ТС — это автобус, такси или HGV и т. д.).

DecryptEriDataRequest

Если данные ERI зашифрованы с помощью общедоступного ключа шифрования регистрирующего органа, который поручил компонент ERI, содержащий данные ERI, и если местному органу не поручено расшифровать эти данные, местный орган власти должен воспользоваться иностранным BOE для их расшифрования.

DecryptEriDataResponse

По соображениям безопасности результат дешифрования транзакции данных ERI будет зашифрован с помощью ключа общедоступного шифрования местных органов власти.

Примечание — Конфиденциальность информации также может быть защищена другими способами, например используя зашифрованное (частная виртуальная схема) соединение.

GetEriHolderRequest

Если иностранное ТС нарушает полосу пропускания, местные органы власти обращаются в местный регистрирующий орган за информацией о владельце ERT.

GetEriHolderResponse

По соображениям безопасности результат транзакции владельца ERT будет зашифрован с помощью ключа общедоступного шифрования местных органов власти.

Примечание — Конфиденциальность информации также может быть защищена другими способами, например используя зашифрованное (частная виртуальная схема) соединение.

Штраф

Иностранное ТС, которое не останавливается по требованию на месте, подвергается наложению штрафа согласно международному праву и/или двусторонним соглашениям.

B.1.16.11 Требования

При входе или проезде в полосе движения должно быть обнаружено местонахождение ТС вблизи движущихся местных и иностранных ТС.

Примечания

1 Так как правила ЕС требуют, чтобы принудительное исполнение было недискриминационным, в ЕС читатели ERI должны, по крайней мере, иметь возможность идентифицировать ТС из других государств-членов.

2 Идентификация иномарок является критичной по времени, если не требуется останавливать эти ТС на месте.

Библиография

[1]	ИСО 3779	Транспорт дорожный. Идентификационный номер автомобилей (VIN). Содержание и структура
[2]	ИСО 7498-2:1989	Системы обработки информации. Взаимодействие открытых систем. Базовая эталонная модель. Часть 2. Архитектура защиты
[3]	ИСО/МЭК 15408 (все части)	Информационные технологии. Методы защиты. Критерии оценки безопасности информационных технологий
[4]	ИСО/МЭК 9798-3:1998	Информационные технологии. Методы защиты. Аутентификация объектов. Часть 3. Механизмы, использующие методы цифровой подписи
[5]	ИСО 14814:2006	Телематика для автомобильного транспорта и дорожного движения. Автоматическая идентификация транспортных средств и оборудования. Эталонная архитектура и терминология
[6]	ИСО 14816	Телематика для дорожного транспорта и дорожного движения. Автоматическая идентификация транспортных средств и оборудования. Нумерация и структура данных
[7]	ИСО/МЭК 10118-3	Информационные технологии. Методы защиты. Хеш-функции. Часть 3. Выделенные хеш-функции
[8]	ИСО/МЭК 15946 (все части)	Информационные технологии. Методы защиты. Криптографические методы на основе эллиптических кривых
[9]	ИСО/МЭК 18033-2	Информационные технологии. Методы защиты. Алгоритмы шифрования. Часть 2. Асимметричные шифры
[10]	ИСО/МЭК 18033-3	Информационные технологии. Методы защиты. Алгоритмы шифрования. Часть 3. Блочные шифры
[11]	ИСО/МЭК 7816-4	Карточки идентификационные. Карточки на интегральных схемах. Часть 4. Организация, безопасность и команды для обмена
[12]	ИСО 14906	Телематика для автомобильного транспорта и дорожного движения. Электронный сбор платежей. Определение прикладного интерфейса для коммуникаций в выделенном коротком диапазоне

УДК 656.13:006.354 ОКС 35.240.60

Ключевые слова: интеллектуальные транспортные системы, электронный сбор платы за проезд, архитектура систем сбора платы за проезд, бортовое оборудование

БЗ 2—2019/33

Редактор *Л.С. Зимилова*
Технический редактор *В.Н. Прусакова*
Корректор *Л.С. Лысенко*
Компьютерная верстка *Е.А. Кондрашовой*

Сдано в набор 06.02.2019. Подписано в печать 13.02.2019. Формат 60×84⅛. Гарнитура Ариал.
Усл. печ. л. 9,77. Уч.-изд. л. 8,42.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Создано в единичном исполнении ФГУП «СТАНДАРТИНФОРМ»
для комплектования Федерального информационного фонда стандартов,

117418 Москва, Нахимовский пр-т, д. 31, к. 2.

gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru