Международный союз электросвязи

МСЭ-Т J.1014

(04/2020)

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

СЕРИЯ J: КАБЕЛЬНЫЕ СЕТИ И ПЕРЕДАЧА СИГНАЛОВ ТЕЛЕВИЗИОННЫХ И ЗВУКОВЫХ ПРОГРАММ И ДРУГИХ МУЛЬТИМЕДИЙНЫХ СИГНАЛОВ

Условный доступ и защита – Заменяемые встроенные решения для обеспечения условного доступа и управления цифровыми правами

Встроенный общий интерфейс для заменяемых решений CA/DRM; усовершенствованная система безопасности – ориентированные на ECI функциональные возможности

Рекомендация МСЭ-Т J.1014

ITU-T

Международный союз электросвязи

Рекомендация МСЭ-Т J.1014

Встроенный общий интерфейс для заменяемых решений CA/DRM; усовершенствованная система безопасности – ориентированные на ECI функциональные возможности

Резюме

Рекомендация МСЭ-Т J.1014 является частью состоящей из нескольких частей документации, охватывающей спецификацию ориентированных на ECI функциональных возможностей **усовершенствованной системы безопасности встроенного общего интерфейса** заменяемых решений для условного доступа/управления цифровыми правами (CA/DRM).

Настоящая Рекомендация МСЭ-Т является переложением стандарта ЕТСИ [b-ETSI GS ECI 001-5-1] и представляет собой результат сотрудничества ИК9 МСЭ-Т и ETSI ISG ECI. Изменения внесены в разделы 1, 3.2, 6.1, 6.2, 6.3 и 8.2.3. Термин "система обработки контента" заменен термином "безопасный видеотракт". Внесены также некоторые редакторские правки.

Хронологическая справка

Издание	Рекомендация	Утверждение	Исследовательская комиссия	Уникальный идентификатор*
1.0	МСЭ-Т J.1014	23.04.2020 года	9-я	11.1002/1000/13575

Ключевые слова

CA, DRM, замена.

* Для получения доступа к Рекомендации наберите в адресном поле вашего браузера URL http://handle.itu.int/, после которого следует уникальный идентификатор Рекомендации. Например, http://handle.itu.int/11.1002/1000/11830-en.

ПРЕДИСЛОВИЕ

Международный союз электросвязи (МСЭ) является специализированным учреждением Организации Объединенных Наций в области электросвязи и информационно-коммуникационных технологий (ИКТ). Сектор стандартизации электросвязи МСЭ (МСЭ-Т) – постоянный орган МСЭ. МСЭ-Т отвечает за изучение технических, эксплуатационных и тарифных вопросов и за выпуск Рекомендаций по ним с целью стандартизации электросвязи на всемирной основе.

На Всемирной ассамблее по стандартизации электросвязи (ВАСЭ), которая проводится каждые четыре года, определяются темы для изучения исследовательскими комиссиями МСЭ-Т, которые, в свою очередь, вырабатывают Рекомендации по этим темам.

Утверждение Рекомендаций МСЭ-Т осуществляется в соответствии с процедурой, изложенной в Резолюции 1 ВАСЭ.

В некоторых областях информационных технологий, которые входят в компетенцию МСЭ-Т, необходимые стандарты разрабатываются на основе сотрудничества с ИСО и МЭК.

ПРИМЕЧАНИЕ

В настоящей Рекомендации термин "администрация" используется для краткости и обозначает как администрацию электросвязи, так и признанную эксплуатационную организацию.

Соблюдение положений данной Рекомендации осуществляется на добровольной основе. Однако данная Рекомендация может содержать некоторые обязательные положения (например, для обеспечения функциональной совместимости или возможности применения), и в таком случае соблюдение Рекомендации достигается при выполнении всех указанных положений. Для выражения требований используются слова "следует", "должен" (shall) или некоторые другие обязывающие выражения, такие как "обязан" (must), а также их отрицательные формы. Употребление таких слов не означает, что от какой-либо стороны требуется соблюдение положений данной Рекомендации.

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

МСЭ обращает внимание на вероятность того, что практическое применение или выполнение настоящей Рекомендации может включать использование заявленного права интеллектуальной собственности. МСЭ не занимает какую бы то ни было позицию относительно подтверждения, действительности или применимости заявленных прав интеллектуальной собственности независимо от того, доказываются ли такие права членами МСЭ или другими сторонами, не относящимися к процессу разработки Рекомендации.

На момент утверждения настоящей Рекомендации МСЭ получил извещения об интеллектуальной собственности, защищенной патентами, которые могут потребоваться для выполнения настоящей Рекомендации. Однако те, кто будет применять Рекомендацию, должны иметь в виду, что вышесказанное может не отражать самую последнюю информацию, и поэтому им настоятельно рекомендуется обращаться к патентной базе данных БСЭ по адресу: http://www.itu.int/ITU-T/ipr/.

СОДЕРЖАНИЕ

Стр.

Введение

Настоящая Рекомендация МСЭ-Т[1] является переложением стандарта ЕТСИ [b-ETSI GS ECI 001-5-1] и представляет собой результат сотрудничества ИК9 МСЭ-Т и ETSI ISG ECI. Изменения внесены в разделы 1, 3.2, 6.1, 6.2, 6.3 и 8.2.3. Термин "система обработки контента" заменен термином "**безопасный видеотракт**"[2]. Внесены также некоторые редакторские правки.

Настоящая Рекомендация призвана способствовать повышению функциональной совместимости и развитию конкуренции в сфере услуг электронной связи, в частности на рынке радиовещательных и аудиовизуальных устройств. В зависимости от ситуации в Государствах-Членах это может быть целесообразно и полезно и в отношении других существующих технологий.

Защита услуг и контента, реализуемая в системах условного доступа (CA) и управления цифровыми правами (DRM), крайне важна в условиях динамичного развития сферы цифрового телерадиовещания и широкополосной связи, охватывающей контент, услуги, сети и оборудование в помещении клиента (CPE), для защиты бизнес-моделей владельцев контента, операторов сетей и операторов систем платного ТВ. Для потребителей особенно важен тот факт, что они смогут использовать ранее приобретенные устройства CPE, например после переезда или смены поставщика сетевых услуг, и даже использовать те же устройства для работы с другими коммерческими видео-порталами. Этого можно достичь путем внедрения взаимодействующих механизмов CA и DRM внутри CPE на основе подходящей архитектуры безопасности.

Настоящая Рекомендация определяет в рамках архитектуры безопасности систему обработки данных для аутентификации и проверки защищенного медиаконтента и образов программного обеспечения, подлежащих обработке внутри ECI-совместимого CPE. Ядром архитектуры безопасности служит **блок лестницы ключей**, который поддерживает безопасную обработку с помощью секретных ключей, нацеливает ключи на конкретные чипы и выполняет аутентификацию происхождения материала ключей.

В разделе 6 приведен обзор системной архитектуры, определены правила обеспечения **устойчивости** для отражения атак и описана связь между элементами архитектуры безопасности, **хостом ECI** и **клиентами ECI**.

В разделе 7 описаны приложения, с которыми можно применять **блок лестницы ключей**, а также соответствующие функции.

Для правильной работы системе обработки данных по безопасности требуется информация о состоянии каждого загруженного **клиента ECI**. Поскольку часть этой информации о состоянии должна носить закрытый характер, она обрабатывается с помощью сегмента повышенной безопасности. Такой сегмент, который необходимо защитить от злонамеренных изменений, **хост ECI** назначает каждому **клиенту ECI**. В разделе 8 даны определение этого сегмента и его конфигурация для нескольких операций, таких как дешифрование или экспорт контента.

В **ECI**-совместимом CPE контент может быть дешифрован, перенаправлен на стандартные выходы, если это разрешено, и повторно зашифрован для экспорта. В разделе 9 описаны способы применения сегмента повышенной безопасности для этих операций.

За аутентификацию элементов отвечает **подсистема обработки сертификатов**, реализованная как специальная функция сегмента повышенной безопасности. В разделе 10 приведены правила аутентификации.

В системе **ECI** используется механизм загрузчика, который позволяет **клиентам ECI** безопасно проверять версию загружаемых учетных данных **хоста ECI** и **клиента ECI** для выявления любой известной проблемы безопасности. Механизм загрузчика опирается на принципы **устойчивости**, описанные в разделе 11.

В разделе 12 указаны ограничения по времени для операций, описанных в настоящей Рекомендации.

---

[1] В Дополнении II определен ряд тематических областей для доработки.

[2] Полужирным шрифтом в тексте настоящей Рекомендации выделены термины, определения которых в контексте встроенного общего интерфейса могут отличаться от общеупотребительных.

Рекомендация МСЭ-Т J.1014

Встроенный общий интерфейс для заменяемых решений CA/DRM; усовершенствованная система безопасности – ориентированные на ECI функциональные возможности

1 Сфера применения

В настоящей Рекомендации описана устойчивая подсистема обработки данных по безопасности для **ECI**, называемая **усовершенствованной системой безопасности**. **Усовершенствованная система безопасности** составляет безопасную основу для аутентифицируемых и загружаемых программных элементов, выполняет вычисления и проверки в связи с безопасностью и управляет шифрованием/дешифрованием контента и обменом контентом с соблюдением соответствующих прав и обязательств. В **усовершенствованной системе безопасности** используется **блок лестницы ключей ECI** [ITU-T J.1015]. По вопросам выполнения вычислений, связанных с безопасностью, см. также [b-ETSI GS ECI 001-5-2].

2 Справочные документы

Указанные ниже Рекомендации МСЭ-Т и другие источники содержат положения, которые путем ссылки на них в данном тексте составляют положения настоящей Рекомендации. На момент публикации указанные издания были действующими. Все Рекомендации и другие источники могут подвергаться пересмотру; поэтому пользователям данной Рекомендации предлагается изучить возможность применения последнего издания Рекомендаций и других источников, перечисленных ниже. Список действующих в настоящее время Рекомендаций МСЭ-Т регулярно публикуется. Ссылка на документ, приведенный в настоящей Рекомендации, не придает ему как отдельному документу статуса Рекомендации.

[ITU-T J.1010] Рекомендация МСЭ-Т J.1010 (2016 год), *Встроенный общий интерфейс для заменяемых решений CA/DRM; сценарии использования и требования.*

[ITU-T J.1011] Рекомендация МСЭ-Т J.1011 (2016 год), *Встроенный общий интерфейс для заменяемых решений CA/DRM; архитектура, определения и обзор.*

[ITU-T J.1012] Рекомендация МСЭ-Т J.1012 (2020 год), *Встроенный общий интерфейс для заменяемых решений CA/DRM; хранилище, загрузчик, интерфейсы, аннулирование объектов CA/DRM.*

[ITU-T J.1013] Рекомендация МСЭ-Т J.1013 (2020 год), *Встроенный общий интерфейс для заменяемых решений CA/DRM; виртуальная машина.*

[ITU-T J.1015] Рекомендация МСЭ-Т J.1015 (2020 год), *Встроенный общий интерфейс для заменяемых решений CA/DRM; усовершенствованная система безопасности – блок лестницы ключей.*

[ETSI ETR 289] ETSI ETR 289 (CSA1/2): 1996, *Digital Video Broadcasting (DVB); Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems.* www.etsi.org/deliver/etsi_etr/200_299/289/01_60/etr_289e01p.pdf

[ETSI TS 100 289] ETSI TS 100 289 (V1.2.1) (CSA3): 2014, *Digital Video Broadcasting (DVB); Support for use of the DVB Scrambling Algorithm version 3 within digital broadcasting systems*. https://www.etsi.org/deliver/etsi_ts/100200_100299/100289/01.02.01_60/ts_100289v010201p.pdf

[ETSI TS 103 127] ETSI TS 103 127 (V1.1.1) (CISSA): 2013, *Digital Video Broadcasting (DVB); Content Scrambling Algorithms for DVB-IPTV Services using MPEG2 Transport Streams*. https://www.etsi.org/deliver/etsi_ts/103100_103199/103127/01.01.01_60/ts_103127v010101p.pdf

[ISO/IEC 9899] ISO/IEC 9899:2011, *Information technology – Programming languages – C*. https://www.iso.org/standard/57853.html

[ISO/IEC 23001-7] ISO/IEC 23001-7:2016, *Information technology – MPEG systems technologies – Part 7: Common encryption in ISO base media file format files*. https://www.iso.org/standard/68042.html

[ISO/IEC 23009-4] ISO/IEC 23009-4:2013, *Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 4: Segment encryption and authentication*. https://www.iso.org/standard/62122.html

[NIST FIPS 180-4] NIST FIPS PUB 180-4:2015, *Secure Hash Standard (SHS)*. https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=910977

[NIST 800-90Ar1] NIST Special Publication 800-90A Rev.1:2015, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final

3 Определения

3.1 Термины, определенные в других документах

Отсутствуют.

3.2 Термины, определенные в настоящей Рекомендации

В настоящей Рекомендации используются следующие термины.

3.2.1 Усовершенствованная система безопасности (Advanced Security System – AS System) – функция устройства **CPE**, соответствующего спецификации **ECI**, обеспечивающая повышенный уровень безопасности (оборудования и программного обеспечения) для **клиента ECI**.

3.2.2 Сегмент AS (AS Slot) – ресурсы блока усовершенствованной системы безопасности, предоставляемые **хостом ECI** исключительно **клиенту ECI**.

3.2.3 AS-API – прикладной программный интерфейс между **клиентом ECI** и его **хостом ECI**, позволяющий **клиенту ECI** обмениваться информацией со своим **сегментом AS** и выполнять в нем операции.

3.2.4 Механизм аутентификации (Authentication Mechanism) – функция **блока лестницы ключей**, определенная в [ITU-T J.1015], которая позволяет **сегменту AS** предоставлять приложения с ключом безопасности для целей, отличных от дешифрования и шифрования контента, таких как аутентификация.

3.2.5 Объект-брат (Brother) – другой **дочерний объект** того же **родительского объекта**.

ПРИМЕЧАНИЕ. – **Родительский, дочерний объекты** и **объект-брат** – это **объекты**, управляющие **сертификатами**.

3.2.6 Сертификат (Certificate) – структура данных, определенная в разделе 5 [ITU-T J.1012], см. также [b-ETSI GS ECI 001-3], с дополнительной защищенной цифровой подписью, которая идентифицирует **объект**.

ПРИМЕЧАНИЕ. – Владелец секретного ключа подписи подтверждает правильность данных (аутентифицирует их), подписывая их своим секретным ключом. Его открытый ключ может использоваться для верификации данных.

3.2.7 Цепочка сертификатов (Certificate Chain) – список **сертификатов**, аутентифицирующих друг друга вплоть до корневого **списка аннулированных сертификатов** и включая его.

ПРИМЕЧАНИЕ. – Как правило, **сертификаты ECI** сопровождаются **списком аннулированных сертификатов**, содержащим **сертификаты**, которые не удается проверить.

3.2.8 Подсистема обработки сертификатов (Certificate Processing Subsystem – CPS) – подсистема **хоста ECI**, обеспечивающая проверку **сертификатов** и дополнительную **устойчивость** защиты к несанкционированному доступу.

3.2.9 Дочерний объект, объекты (Child, Children) – **объект (объекты)**, указанный (указанные) в **сертификате**, подписанном (общим) **родительским объектом**.

ПРИМЕЧАНИЕ. – **Родительский, дочерний объекты** и **объекты-братья** – это объекты, управляющие **сертификатами**: данные инициализации и программное обеспечение, которое используется для запуска системы SoC оборудования **CPE**.

3.2.10 Свойства контента (Content Properties – CP) – свойства контента, предоставляющие информацию о правах и обязательствах, связанных с последующим применением или преобразованиями контента, например сведения о правах на использование, выборочное управление выходом и сведения о родительском контроле.

3.2.11 ECI (Embedded CI) – архитектура и система, определенные в ISG ETCИ "Встроенный CI", позволяющие разрабатывать и внедрять программно заменяемые **клиенты ECI** в оборудовании в помещении клиента (CPE), тем самым обеспечивая совместимость устройств CPE с интерфейсом **ECI**.

3.2.12 Клиент ECI (Embedded CI Client) – практическая реализация клиента CA/DRM, соответствующая спецификациям **ECI**.

3.2.13 Загрузчик клиента ECI (ECI Client Loader) – функция **хоста ECI**, использующая **систему AS** только для обеспечения функционирования, проверки и установки нового образа программного обеспечения **клиента ECI** в контейнере **ECI хоста ECI**.

3.2.14 Экосистема ECI (ECI Ecosystem) – коммерческая эксплуатация, охватывающая **доверительный орган** и несколько платформ, а также установленное на местах оборудование **CPE**, поддерживающее интерфейс **ECI**.

3.2.15 Хост ECI (ECI Host) – аппаратная и программная система в составе CPE, которая охватывает функциональные возможности, связанные с **ECI**, и включает интерфейсы для связи с **клиентом ECI**.

ПРИМЕЧАНИЕ. – **Хост ECI** входит в состав встроенного программного обеспечения CPE.

3.2.16 Загрузчик хоста ECI (ECI Host Loader) – функция начальной загрузки CPE, использующая **систему AS** только для выполнения проверки и установки программного обеспечения **хоста ECI** в CPE.

ПРИМЕЧАНИЕ. – В конфигурации многоступенчатой загрузки этот термин используется для обозначения всех критически важных функций загрузки, связанных с загрузкой **хоста ECI**.

3.2.17 Корневой ключ ECI (ECI Root Key) – открытый ключ, который служит источником аутентификации сертифицированных объектов **ECI** и **сертификатов**.

3.2.18 Объект (Entity) – организация (например, производитель, **оператор** или **поставщик средств безопасности**) или объект реального мира (например, **хост ECI**, **система управления платформой** или **клиент ECI**), идентифицируемые уникальным идентификатором в **экосистеме ECI**.

3.2.19 Соединение экспорта (Export Connection) – аутентифицированное отношение между **сегментом AS**, дешифровывающим контент, и **сегментом AS**, который впоследствии заново зашифровывает дешифрованный контент, указывающее, что такое повторное шифрование разрешено.

3.2.20 Родительский объект (Father) – сторона, подписавшая **сертификат дочернего объекта**.

ПРИМЕЧАНИЕ. – **Родительский**, **дочерний объекты** и **объект-брат** – это **объекты**, управляющие **сертификатами**.

3.2.21 Лестница ключей (Key Ladder) – функция **блока лестницы ключей**, определенного в [ITU-T J.1015], предназначенная для вычисления управляющих слов и связанной с ними информации по использованию управляющих слов для применения в функции дешифрования или повторного шифрования контента CPE.

3.2.22 Блок лестницы ключей (Key Ladder Block) – устойчивый механизм обеспечения безопасности для вычислений при дешифровании, шифровании и аутентификации ключей, определенный в [ITU-T J.1015], включающий **лестницу ключей** и **механизм аутентификации**.

3.2.23 Микроклиент (Micro Client) – **клиент ECI** или не **клиент ECI**, способный дешифровывать контент, перешифрованный с помощью **микросервера**.

3.2.24 Система Micro DRM (Micro DRM System) – система защиты контента, которая с помощью **микросервера** перешифровывает контент в CPE и позволяет аутентифицированным **микроклиентам** дешифровывать этот перешифрованный контент.

ПРИМЕЧАНИЕ. – **Микросервер** и **микроклиенты** предоставляются **оператором системы Micro DRM**.

3.2.25 Микросервер (Micro Server) – **клиент ECI**, способный импортировать дешифрованный контент, перешифровывать этот контент и аутентифицировать конкретный **клиент ECI** или группу **клиентов ECI** в качестве **цели** для последующего дешифрования.

3.2.26 Оператор (Operator) – организация, выполняющая роль **системы управления платформой** и обладающая правом подписи в **экосистеме ECI**, выданным **доверительным органом ECI**.

ПРИМЕЧАНИЕ. – **Оператор** может осуществлять управление несколькими **платформами**.

3.2.27 Система управления платформой (Platform Operation – PO) – конкретный образец процесса оказания технической услуги, обладающего едиными идентификационными данными **ECI** с точки зрения безопасности.

3.2.28 Сервер подготовки (Provisioning Server) – сервер, обычно расположенный в защищенном помещении, который предоставляет ключи и другую защищенную информацию для выполнения функций шифрования или дешифрования посредством **сегмента AS**.

3.2.29 Аннулирование (Revocation) – статус исключения **объекта** ввиду его внесения в **список аннулированных сертификатов**.

3.2.30 Список аннулированных сертификатов (Revocation List – RL) – список **сертификатов**, которые были аннулированы и, следовательно, не подлежат дальнейшему использованию.

3.2.31 Устойчивость (Robustness) – характеристика определенной функции безопасности **ECI**, отражающая трудоемкость и/или стоимость усилий, направленных на нарушение защиты, реализуемой данной функцией.

3.2.32 Корневой сертификат (Root Certificate) – надежный сертификат, служащий источником аутентификации цепочки сертификатов в **экосистеме ECI**.

3.2.33 Безопасный видеотракт (Secure Video Path) – все функции оборудования CPE, в рамках которых производится обработка и необходимое для обработки временное хранение контента, начиная с дешифрования контента и заканчивая его повторным шифрованием с помощью микроклиента или системы защиты выходных данных.

3.2.34 Поставщик услуг по обеспечению безопасности (Security Vendor) – компания, предоставляющая системы безопасности **ECI**, включая **клиенты ECI** для **операторов системы управления платформой**.

3.2.35 Цель (Target) – **микроклиент** или группа **микроклиентов**, для которых контент перешифровывается **микросервером**.

4 Сокращения и акронимы

В настоящей Рекомендации используются следующие сокращения и акронимы.

ACF	Advanced Security Control Field	Поле контроля повышенной безопасности
AD	Associated Data	Связанные данные
AES	Advanced Encryption Standard	Усовершенствованный стандарт шифрования
AK	Authentication Key	Ключ аутентификации
API	Application Programming Interface	Прикладной программный интерфейс
ARK	Advanced Security Random Key	Случайный ключ повышенной безопасности
AS	Advanced Security	Повышенная безопасность
CA	Conditional Access	Условный доступ
CBC	Cypher Block Chaining	Сцепление блоков шифра

CENC	Common Encryption	Общепринятая система шифрования
CISSA	Common IPTV Software-oriented Scrambling Algorithm	Общий алгоритм скремблирования, ориентированный на программное обеспечение IPTV
CP	Content Properties	Свойства контента
CPE	Customer Premises Equipment	Оборудование в помещении клиента
CPS	Certificate Processing Subsystem	Подсистема обработки сертификатов
CSA	Common Scrambling Algorithm	Общий алгоритм скремблирования
CTR	Counter Mode	Режим счетчика
CW	Control Word	Управляющее слово
DRM	Digital Rights Management	Управление цифровыми правами
EAC	Export Authorization Certificate	Сертификат разрешения на экспорт
EAOC	Export Authorization **Operator Certificate**	**Сертификат оператора**, выдающего разрешение на экспорт
ECI	Embedded Common Interface	Встроенный общий интерфейс
ECM	Entitlement Control Message	Сообщение управления правами доступа
ECP	Enhanced Content Protection	Система расширенной защиты контента
EGC	Export Group Certificate	Сертификат группы по экспорту
ERC	Export **Revocation** Certificate	Сертификат **аннулирования** разрешения на экспорт
ESC	Export System Certificate	Сертификат системы экспорта
LK	Link Key	Ключ связи
MPEG	Moving Picture Experts Group	Группа экспертов по движущимся изображениям
MSCSK	Micro Server Chipset Secret Key	Секретный ключ чипсета микросервера
PES	Packetized Elementary Stream	Пакетированный элементарный поток
PO	Platform **Operation**	**Система управления** платформой
POC	**Platform Operation** Certificate	Сертификат **системы управления платформой**
POPK	**Platform Operation** Public Key	Открытый ключ **системы управления платформой**
REAOC	**Revocation** Export Authentication **Operator** Certificate	Сертификат **оператора** проверки подлинности **аннулирования** разрешения на экспорт
REE	Rich Execution Environment	Функционально богатая среда выполнения
RFU	Reserved for Future Use	Зарезервировано для использования в будущем
RK	Random Key	Случайный ключ
RL	**Revocation List**	**Список аннулированных сертификатов**
SPK	Sender Public Key	Открытый ключ отправителя
TA	Trust Authority	Доверительный орган
TEE	Trusted Execution Environment	Доверенная среда выполнения
TLS	Transport Layer Security	Безопасность транспортного уровня
TPEGC	Third Party Export Group Certificate	Сертификат сторонней группы по экспорту
TS	MPEG 2 Transport Stream	Транспортный поток MPEG 2

URI	Usage Rights Information	Информация о правах на использование
XT	eXTension field	Поле расширения

5 Соглашения

Использование в настоящей Рекомендации терминов, выделенных жирным шрифтом и начинающихся с прописной буквы[3], указывает на то, что эти термины имеют особое значение, связанное с ECI, которое может отличаться от общепринятого значения этих терминов.

6 Основные принципы

6.1 Обзор

Настоящая Рекомендация относится к серии Рекомендаций МСЭ-Т, в которых рассматриваются архитектура **ECI** [ITU-T J.1011] (см. также [b-ETSI GS ECI 001-1]) и базовые требования **ECI** [ITU-T J.1010] (см. также [b-ETSI GS ECI 001-2]).

Основные принципы построения **усовершенствованной системы безопасности** представлены на рисунке 6-1. Ядро **усовершенствованной системы безопасности** составляет определенный в [ITU-T J.1015] **блок лестницы ключей**, который позволяет выполнять безопасную обработку с помощью секретных ключей, нацеливать ключи на специальные чипы и определять происхождение материала ключей.

Для загрузки образов используется **ядро загрузчика**. Проверка **ECI**-статуса образов **хоста ECI**, образов **клиента ECI** и учетных данных **системы управления платформой** (PO) осуществляется с помощью **подсистемы обработки сертификатов** с использованием последнего **корневого ключа ECI** и корневого **списка аннулированных сертификатов ECI**. Номера версий **корневого ключа ECI** и **списка аннулированных сертификатов ECI**, используемых **хостом ECI** и другими **клиентами ECI**, проверяются загруженными **клиентами ECI**. Они могут отказаться дескремблировать контент в случае обнаружения неприемлемых версий в соответствии с принципом **контроля за исполнением аннулирования сертификатов ECI**. Зашифрованные образы **клиентов ECI** дешифруются при загрузке.

Каждый **клиент ECI** использует сегмент повышенной безопасности. **Сегмент AS** идентифицируется открытым ключом **системы управления платформой ECI**. **Хост ECI** гарантирует, что взаимодействие с **клиентом ECI** через **AS-API** осуществляется посредством **сегмента AS**, выделенного этому **клиенту ECI**. Каждый **сегмент AS** описывается состоянием сегмента и состоянием сеанса для данной операции шифрования/дешифрования. **Сегмент AS** может использоваться для целей дешифрования или шифрования. Состояние сеанса **сегмента AS** также включает конфигурацию (config), определяющую детали операции и способ определения состояния сеанса. **Клиент ECI** предоставляет информацию о конфигурации и исходные данные для другой информации о состоянии. **Блок лестницы ключей** используется для аутентификации открытого ключа отправителя (SPK), открытого ключа системы управления платформой (POPK) и информации о конфигурации. **Сегмент AS** может подавать случайные числа на выбранные входы **блока лестницы ключей** для генерирования случайных ключей или использования в качестве одноразового кода, чтобы обеспечить **блок лестницы ключей** последними вычисленными исходными данными. Этот механизм можно использовать для того, чтобы предотвращать повторное воспроизведение зашифрованного контента и обеспечить постоянное в режиме онлайн предоставление **сервером подготовки сегмента AS**.

При дешифровке контента можно определить **свойства контента** и вычислить управляющие слова, создав тем самым прочную связь с дешифрованным контентом. **Свойства контента** направляются вместе с контентом на любой стандартный выход, с тем чтобы обеспечить надлежащую настройку механизмов защиты для этого выхода. Эти свойства сравниваются со свойствами контента перешифрованного в **соединении экспорта**. **Соединение экспорта** может осуществляться только через соответствующие **цепочки сертификатов** экспорта/импорта, которые проверяются

[3] В русской версии данные термины выделяются только жирным шрифтом. – *Прим. перев.*

подсистемой обработки сертификатов от имени **сегмента AS**. Механизм управления выходами может отключать стандартные выходы.

Вычисленные управляющие слова могут быть синхронизированы с форматированным контентом транспортного потока MPEG посредством протокола чередующихся битов. Для этой цели в безопасном видеотракте используется механизм двойной буферизации с текущим/следующим управляющим словом для обработки потока.

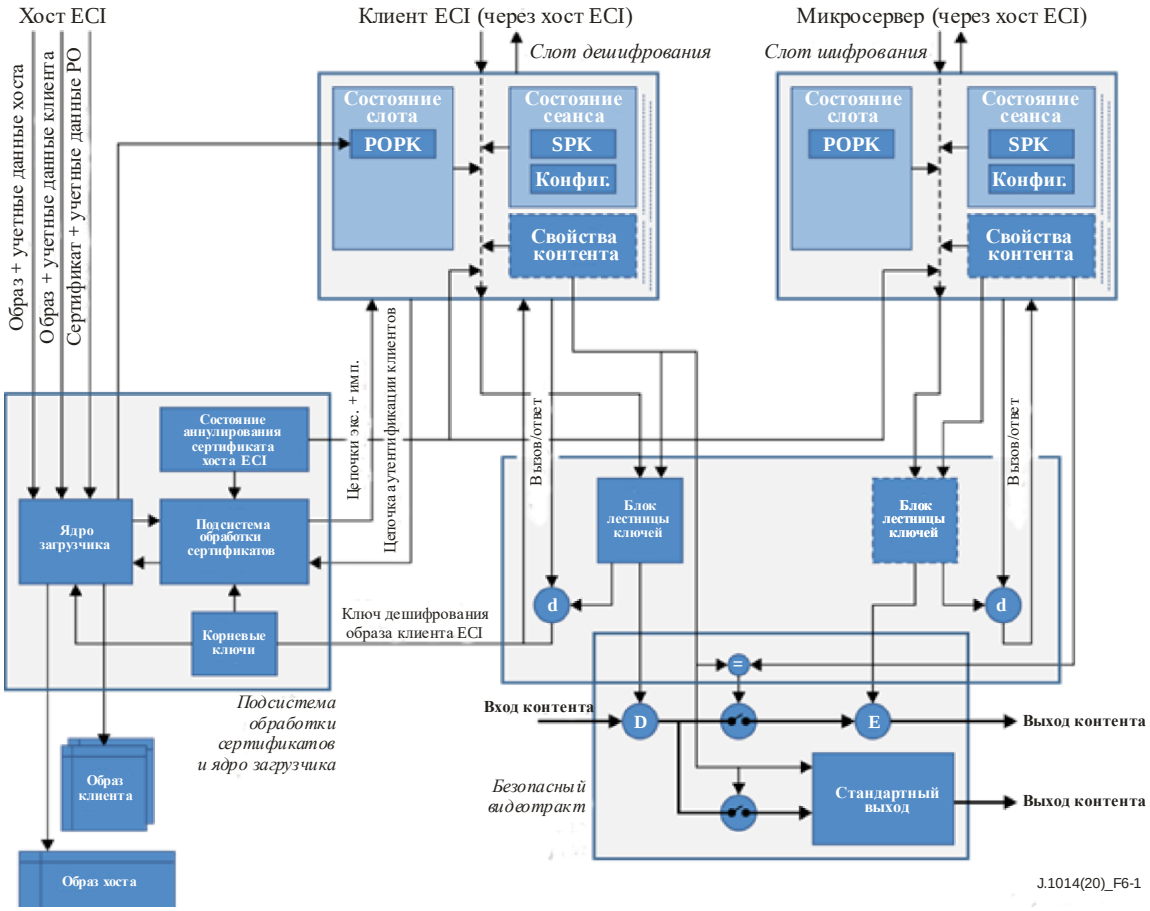

Рисунок 6-1 – Блок-схема усовершенствованной системы безопасности

6.2 Модель обеспечения устойчивости системы

Система AS требует устойчивой реализации. **Устойчивость** обычно измеряется трудоемкостью и/или затратами, необходимыми для того, чтобы обойти все меры безопасности, то есть определить секретные значения или манипулировать состоянием или значениями в защищенной системе.

В настоящей Рекомендации не определяется конкретный режим обеспечения **устойчивости** различных функций **ECI**. Тем не менее архитектура **устойчивости ECI** основана на предпосылке, что некоторые функции устойчивее других. Это показано на рисунке 6-2.

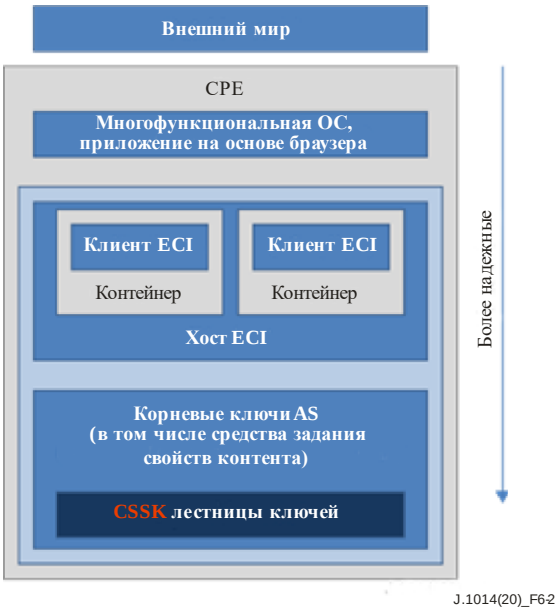

Рисунок 6-2 – Предпосылка устойчивости системы в отношении ECI

Наименее устойчивая среда – это внешний мир, где может существовать любая угроза. Данные, проходящие через эту среду, должны быть защищены от манипуляций и несанкционированной проверки посредством использования методов аутентификации и шифрования. Многофункциональная операционная система (обычно включающая браузер) может быть в какой-то степени устойчивой к манипуляциям и вторжению, но, как правило, она не в состоянии противостоять агрессивным внешним хакерским атакам или атакам с участием пользователя. Функции **клиентов ECI** и **хоста ECI**, важные с точки зрения безопасности, работают в среде, которая хорошо защищена от таких атак. В случае взлома **хоста ECI** под угрозой оказываются и **клиенты ECI**. В дополнение к устойчивости к внешним атакам **клиенты ECI** изолированы с помощью виртуальной машины **ECI** [ITU-T J.1013], см. также [b-ETSI GS ECI 001-4], то есть они не могут получить доступ ни к информации **хоста ECI**, ни к любому другому **клиенту ECI**, кроме как через интерфейсы, определенные API **ECI**. **Хост ECI** также гарантирует **клиентам ECI** доступ к **усовершенствованной системе безопасности** и **блоку лестницы ключей**. В ядре **блока лестницы ключей** находится **секретный ключ чипсета**, позволяющий обращаться только к каждому отдельному **CPE ECE**. Как правило, **блок лестницы ключей** и основные части **усовершенствованной системы безопасности** реализуются аппаратно и/или с помощью высокоустойчивого встроенного ПО.

На практике уровень устойчивости компонентов системы обеспечения безопасности представляет собой величину из некоторого континуума, но с точки зрения архитектуры эти компоненты можно классифицировать по иерархическому принципу. В контексте ECI их можно описать следующим образом:

0. Внешний мир (за пределами устройства) характеризуется уровнем устойчивости 0. Поэтому **клиенту ECI** необходим защищенный протокол для связи с головной станцией. Этот вопрос выходит за рамки Рекомендации, посвященной **ECI**.

1. Операционная система (ОС), драйверы, приложения и браузер устройства характеризуются уровнем устойчивости 1. Среду, в которой выполняется программный код этих компонентов, часто называют функционально богатой средой выполнения (REE). Она включает в себя большую часть прикладного и системного программного обеспечения, а также драйверов. Ввиду большого объема и обширной функциональности этого программного обеспечения в него приходится часто вносить исправления для устранения уязвимостей. **Хост ECI** и **клиент ECI** могут пользоваться предоставляемыми REE услугами (например, услугами сетей на базе протокола TLS), но не могут полагаться на эту среду в деле обеспечения безопасности, будь то шифрование или аутентификация конечных точек.

2. На уровне устойчивости 2 выполняются функции **хоста ECI**, виртуальной машины и **клиента ECI**. Все они должны работать в защищенной среде выполнения, нередко на одном и том же

процессоре. На этом уровне выполняется только аутентифицированный код с аппаратной изоляцией памяти и устройства от REE, в том числе от ее ядра и драйверов. Такую среду иногда называют доверенной средой выполнения (TEE). Данный уровень устойчивости может, в числе прочего, реализовываться на специальном процессоре или процессоре системы безопасности с изолированной памятью.

3. Для работы **безопасного видеотракта** требуется более высокий уровень устойчивости, чем для **хоста ECI** и **клиента ECI** – уровень 3. Он может реализовываться на основе сочетания защищенного аппаратного и защищенного микропрограммного обеспечения.

4. Самые защищенные и надежные компоненты – это **лестница ключей**, начальный загрузчик и подсистемы обработки и аннулирования сертификатов, работающие на уровне устойчивости 4. Лучше всего встраивать их в защищенное аппаратное обеспечение, но ввиду особенностей содержания Рекомендации по **ECI** для работы некоторых их частей может потребоваться микропрограммное обеспечение, выполняющееся на специальном процессоре системы безопасности. На этом уровне необходимо принимать конкретные меры для предотвращения раскрытия секретной информации из усовершенствованной системы безопасности и лестницы ключей, а также обеспечения секретности производимых в них вычислений.

6.3 Безопасный видеотракт и управление системой защиты выходных данных

После того как контент дешифрован, от несанкционированного использования его защищает **безопасный видеотракт**. В **безопасном видеотракте** над незашифрованным контентом могут выполняться различные операции по обработке видео (и звука): демультиплексирование, декодирование, масштабирование, повторное кодирование, вставка водяных знаков и т. д. Все операции, их составные части (включая работу с временной памятью и шинами, используемыми для передачи контента между различными подсистемами оборудования CPE) и интерфейсы управления (свойства контента и, возможно, проприетарные или специфичные для конкретной реализации интерфейсы), управляющие критически важными этапами обработки свойств контента и параметрами защиты выходных данных, считаются частью **безопасного видеотракта**.

Свойства контента [ITU-T J.1012] определяют, какие операции разрешены и какие выходы, системы защиты выходных данных и параметры обеспечивают достаточную защиту контента. **Усовершенствованная система безопасности** предоставляет криптографический механизм, называемый аутентификацией защиты контента (см. пункт 8.2.3), который позволяет неявно аутентифицировать свойства контента при вычислении контрольного слова, используя блок лестницы ключей. Система защиты выходных данных может выбирать настройки аутентификации свойств контента. **Свойства контента**, не аутентифицированные неявно функцией аутентификации свойств контента, имеют уровень устойчивости **клиента ECI** и **хоста ECI**.

Следует отметить, что для достаточно подробного описания устойчивой реализации **безопасного видеотракта** и систем защиты выходных данных потребуются дополнительные спецификации.

На рисунке 6-3 подробно изображена логика обеспечения безопасности при доставке **свойств контента**. Сначала **клиент ECI** устанавливает **свойства контента** в данном сеанса для следующего раздела контента, подлежащего декодированию. Затем **клиент ECI** предоставляет входные данные для вычисления контрольного слова, включая Field1 (в составе входного вектора **elk**: см. пункты 7.1, 8.2.3, 8.2.4.7). Поля свойств контента Field1 выбираются в соответствии с первыми двумя байтами Field1 в функции аутентификации свойств контента затем неявно аутентифицируются с использованием этого значения в качестве входного ключа симметричного шифрования для вычисления управляющего слова (CW) в лестнице ключей. Неверное значение входного ключа симметричного шифрования дает неверное значение CW, из-за чего дешифрование контента будет невозможным. Далее значения Field1 сравниваются SVP со значениями соответствующих свойств контента, установленными **клиентом ECI**. В случае каких-либо расхождений вывод контента невозможен. Вывод через конкретную систему защиты выходных данных может также быть заблокирован путем установки соответствующего значения поля управления выводом.

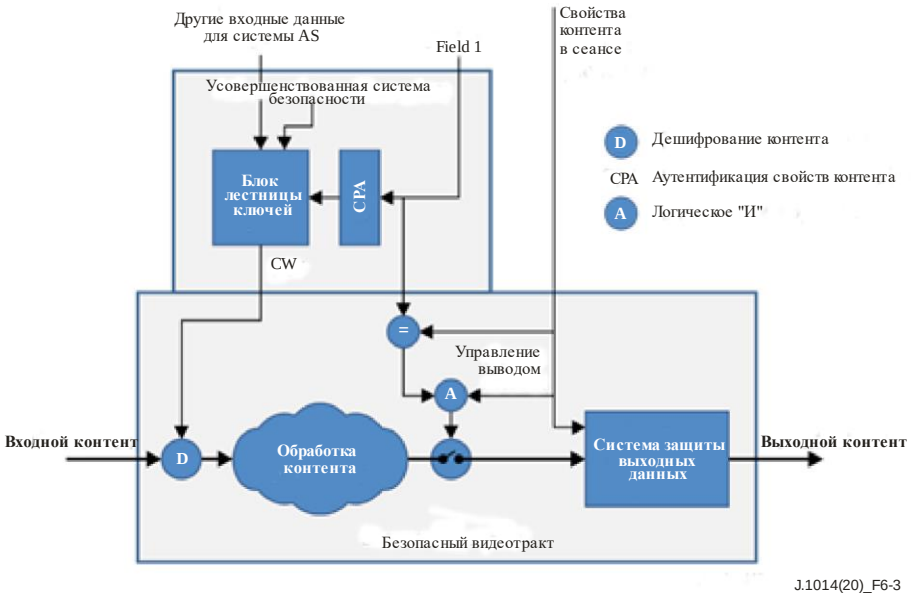

Рисунок 6-3 – Безопасный видеотракт и аутентификация свойств контента

6.4 Принципы описания

6.4.1 Свобода при реализации

В настоящей Рекомендации определены состояния и функции, действующие в **системе AS**, которые приводят к ее новому состоянию. В настоящей Рекомендации не определяются конкретные представления состояния реализации, и они могут полностью определяться реализацией при условии, что ее поведение может быть воссоздано вплоть до состояний и переходных последовательностей с использованием функций перехода, определенных в настоящей Рекомендации.

ПРИМЕЧАНИЕ. – Во многих случаях существенной частью функции перехода является функция **лестницы ключей**, определенная в [ITU-T J.1015].

Например, реализация AS может иметь быструю подсистему обработки сертификатов (CPS), способную повторно аутентифицировать POPK из **цепочки сертификатов системы управления платформой** при каждом применении **блока лестницы ключей**. В этом случае **сегменту AS** не нужно хранить POPK в качестве аутентифицированного значения защищенным от несанкционированного доступа способом. Аналогично, в некоторых случаях реализации LK_1 (симметричный ключ верхнего уровня в **лестнице ключей**) для **сегмента AS** может вычисляться один раз с использованием двух асимметричных криптографических операций, тогда как в других случаях реализации, способной достаточно быстро выполнять асимметричные криптографические операции, LK_1 может вычисляться заново по первоначальным исходным данным каждый раз, когда применяется **лестница ключей**.

6.4.2 Стиль описания и отношение к AS-API

Не существует прямого API между **клиентом ECI** и **усовершенствованной системой безопасности**. В качестве канала связи выступает **хост ECI**. Тем не менее определения операций в **сегменте AS** непосредственно отражаются в сообщениях **AS API**, как определено в [ITU-T J.1012], за исключением параметра slotId, который не требуется для сообщений AS API **клиента ECI**. Параметр slotId предоставляется **системе AS хостом ECI**.

Транзакции **хоста ECI** (от имени **клиента ECI**) в **сегменте AS** определяются как декларации C-функций, которые описывают элементарную операцию над состоянием **сегмента AS**. Это может привести к новому состоянию сегмента. Конкретное представление параметров функции не оказывает прямого влияния на функциональные возможности, описанные в настоящей Рекомендации, за исключением случаев, относящихся к криптографическим функциям. Однако представление важно для определения **AS API** в [ITU-T J.1012].

7 Применение лестницы ключей и связанные с этим функции

7.1 Общие положения

Центральную роль во всех устойчиво реализованных вычислениях секретных ключей в **CPE ECI** играют **лестница ключей** и **механизм аутентификации**, определенные в [ITU-T J.1015]. **Усовершенствованная система безопасности** применяет эти функции, как указано в [ITU-T J.1015], с входными и выходными данными, определенными в настоящей Рекомендации. Всеми входными данными **блока лестницы ключей** управляет **система AS**; соответствующие правила **устойчивости AS** и спецификация **блока лестницы ключей** [ITU-T J.1015] делают невозможным какое бы то ни было наблюдение за этими данными или манипулирование ими.

7.2 Система AS и аутентификация клиентских данных

Усовершенствованная система безопасности может получать данные от **клиента ECI**. **Система AS** предоставляет средства для проверки подлинности этих данных с использованием входа AD **блока лестницы ключей**.

Система AS вычисляет вход AD **блока лестницы ключей** как хеш дополнительных данных, подлежащих аутентификации, в сочетании с вычислением CW или AK. Следуя системе обозначения битовых строк из [ITU-T J.1015], можно записать:

$$AD = hash(\ ACF \parallel lm \parallel ARK \parallel P_1 \parallel ... \parallel P_m \parallel C_1 \parallel ... \parallel C_m \parallel XT)$$

Функция *hash* определена в пункте A.1. lm – это 8-битовый входной код, содержащий двоичное представление m. Значение m соответствует значению параметра m из определения **блока лестницы ключей**. Длина каждого элемента P_i при переносе открытых ключей (значения POPK) составляет 2048 бит. Длина элемента C_i определена как равная длине структуры SessionConfig в пункте 8.2.2.5, а XT длиной 256 служит механизмом расширения общего назначения. Его значение устанавливается в 0. ARK – это 128-битовое число, призванное представлять случайное значение или все нули в том случае, если ввод случайного числа не требуется. ACF – это управляющая величина, определяющая режим работы.

7.3 Режим асимметричного микросервера

Система AS применяет **механизм аутентификации** для загрузки секретного ключа отправителя **микросервера** в **сегмент AS** в целях асимметричной аутентификации между **микросерверами** и **микроклиентами**.

Общий принцип вычисления для режима асимметричной аутентификации показан на рисунке 7-1.

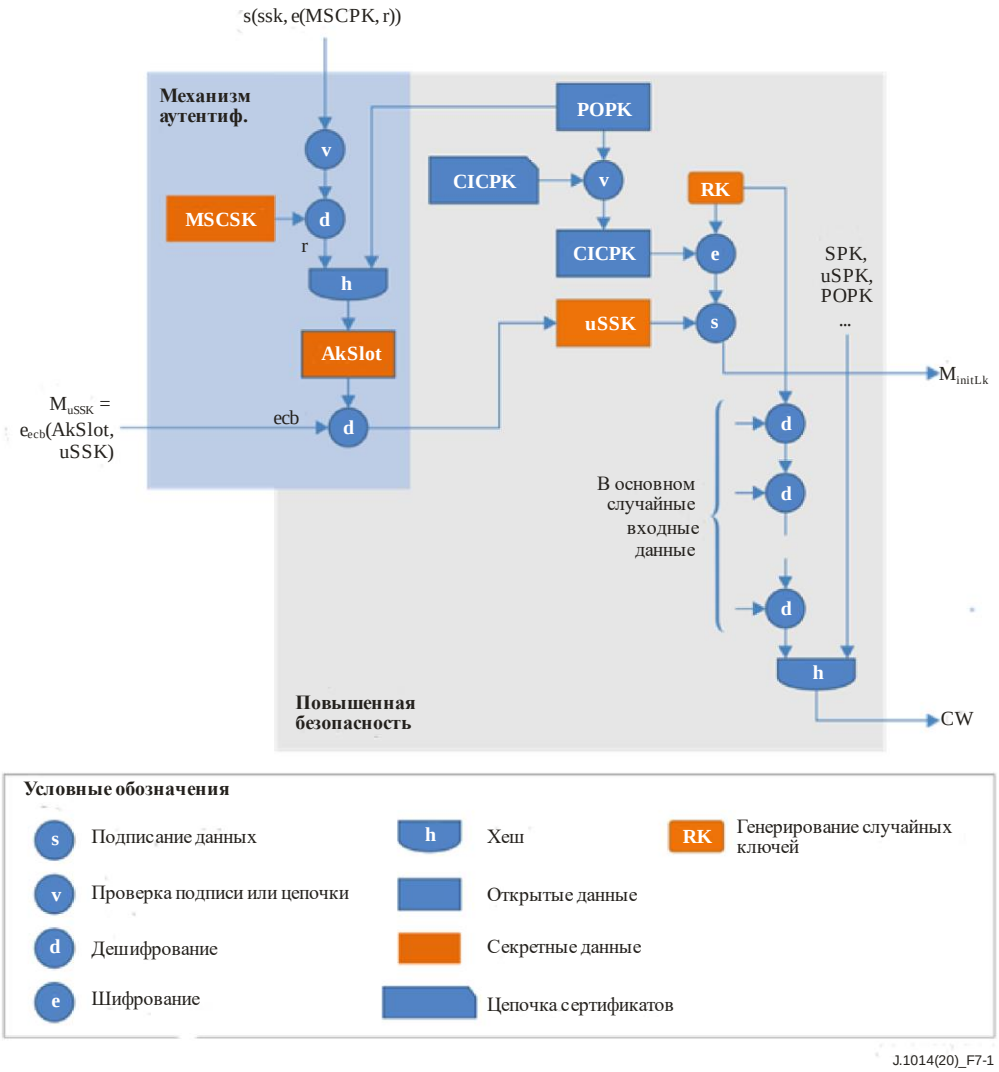

Рисунок 7-1 – Вычисления для режима асимметричного микросервера

Механизм аутентификации используется для вычисления ключа аутентификации **сегмента AS** AkSlot с использованием в том числе ключа чипсета, который называется здесь секретным ключом чипсета **микросервера** (MSCSK). AkSlot используется для загрузки секретного ключа **микросервера** uSSK. Аутентификация открытого ключа чипсета CLCPK **целевого микроклиента** осуществляется с помощью **подсистемы обработки сертификатов** с использованием POPK в качестве корня и **цепочки сертификатов**, содержащей CLCPK в последнем **сертификате** цепочки. Генерируется случайный ключ RK, а для генерации сообщения инициализации M_{initLk} **лестницы ключей микроклиента** используются ключи CLCPK и uSSK. Случайный ключ также используется как симметричный ключ верхнего уровня **лестницы ключей** со структурой и хеш-функцией, определенными в пункте 7.1 [ITU-T J.1015]. Вычисленное управляющее слово CW используется для шифрования **микросервером**. В **микроклиенте** для вычисления CW в целях дешифрования может быть использована стандартная **лестница ключей**.

Специфика вычисления uSSK (функция **ldUssk**) описана в пункте 8.2.4.10.

Вычисление M_{initLk} производится так, как определено в приведенной ниже схеме вычислений, с использованием условных обозначений из пункта 7.1 [ITU-T J.1015]:

- Mkey = cl-chipset-ID || E(ClCPK,LK);
- M_{initLk} =(Mkey || S(uSSK,Mkey)),
- где || – функция побитовой конкатенации; cl-chipset-ID – идентификатор чипсета CPE; E() – функция асимметричного шифрования; S() – функция асимметричной подписи, как определено в пункте 7.2 [ITU-T J.1015].

При загрузке uSSK **сегмент AS** генерирует новый RK в соответствии с пунктом A.3.

Вычисление CW по LK и его входным данным осуществляется так же, как в **механизме лестницы ключей**, определенном в пункте 7.2 [ITU-T J.1015], с теми же входными данными, какие определены там, и теми же выходными данными (CW, CW-URI), но вместо вычисления LK1 применяется случайный сеансовый ключ RK, сгенерированный **сегментом AS микросервера**.

7.4 Интерфейс с безопасным видеотрактом

Лестница ключей, включающая расширение режима асимметричного **микросервера**, может вычислять управляющие слова с дополнительным CW-URI. Они должны безопасным образом передаваться ресурсу дешифрования или шифрования в подсистеме обработки контента, которая может (временно) хранить информацию CW и CW-URI в сочетании с информацией о синхронизации ключей. Для применений в транспортном потоке информация о синхронизации ключей состоит из бита "текущий/следующий", что указывает на два места хранения значений CW. Для применений на основе файлов доступна одна ячейка CW для ресурса шифрования и дешифрования.

Другая информация, сопровождающая управляющие слова из **сегмента AS** в **безопасный видеотракт**, включает:

- Применение CW в качестве управляющего слова при шифровании или дешифровании.

ПРИМЕЧАНИЕ. – Вместе с CW-URI это представляет собой разрешение на применение CW.

- Информация для аутентификации экспорта.
- Свойства контента.
- Свойство нечетности/четности управляющего слова для режима дескремблирования TS.

7.5 Определение выходных и входных данных блока лестницы ключей AS

В этом разделе описано преобразование входных данных **лестницы ключей** и **механизма аутентификации** с их расширениями, определенными в пунктах 7.2 и 7.3, посредством переменных и структур в стиле языка программирования C. В остальной части настоящей Рекомендации символы имен входных данных используются для определения различных применений.

Преобразование последовательности октетов в структуру языка C происходит в соответствии со следующими правилами (в порядке следования битов начиная с младшего):

- Бит-поля структур преобразуется начиная с первого поля, содержащего младший бит (0) первого октета.
- Структуры длиной, не кратной 8 битам, дополняются с конца до следующего значения, кратного 8 битам, зарезервированными битами, которые устанавливаются в ноль до следующего значения, кратного 8-битам.
- 16-, 32- и 64-битовые объекты преобразуются в порядке следования начиная с младшего значащего байта.
- Массивы преобразуются в порядке возрастания индекса.

Последовательности октетов преобразуются в битовые строки с использованием функции OS2BSP, как определено в [ITU-T J.1015].

ПРИМЕЧАНИЕ. – Вышеприведенные правила обеспечивают, что порядок нумерации битов, используемый для целочисленных значений, представленных переменными в стиле языка C, совпадает с тем, который используется в [ITU-T J.1015] для соответствующих входных данных **блока лестницы ключей**.

Соглашение об именовании переменных приведено в таблице 7-1.

Таблица 7-1 – Соглашение об именовании переменных интерфейса лестницы ключей

Вход или выход блока лестницы ключей	Биты	Соглашение об именовании переменных в стиле языка C
CW-URI	64	ulong **cwUri**;
AD	256	uchar **ad**[32]; /* не используется напрямую, см. пункт 7.2 */
SPK-URI	64	ulong **spkUri**;
SPKi, i=1..16	2048 × 16	typdef uchar PubKey[256]; PubKey **spk**[16]; /* (spk[i-1] == SPKi) */
m		uchar **nSpk**;
Вход V	64 + 2048 + 2048	typedef struct InputV{ ulong chipsetId; uchar elk1[256]; uchar signature[256]; } InputV; InputV **inputV**;
E(LKi,LKi+1), i=1..24; LKt+1=r	256 × 24	typedef uchar SymKey[32]; Symkey **elk**[i]; /* (elk[i-1] == E(LKi,LKi+1)) */ /* **C-input** == E(LKt-1,LKt1), то есть предпоследний вход ***/**
t		uchar **nElk**;
τ_b		Значение устанавливается в 0
Chipset-ID	64	ulong **chipsetId**;
Challenge	128	uchar **challenge**[16];
Response	128	uchar **response**[16];
Входы и выходы, определенные в настоящей Рекомендации, приведены ниже.		
ACF	128-8	uchar **acf**[15]; /* рабочий режим */
ARK	128	uchar **ark**[16];
$\mathbf{P_i}$	2048 × 32	PubKey **pk**[32]; /* применяются первые значения **m** */
$\mathbf{C_i}$	sizeof(SessionConfig) × 32	SessionConfig **config**[?]; /* Переменная SessionConfig определена в пункте 8.2.2.6 */
XT	256	uchar **XT**[32]; /* значение всегда устанавливается в 0 */
M_{initLk}	64+ 2048 + 2048	InputV **mInitLk**;

С использованием приведенных выше входных переменных для получения результатов определяются следующие функции на языке C:

```
SymKey blockV_blockC_KeyLadder(InputV inputV, SymKey spk)
```

Семантика

Эта функция вычисляет функцию блока V и блока C в **лестнице ключей** для получения lk1.

В случае асимметричного сервера сообщение инициализации **целевого микроклиента**, как определено в пункте 7.3, вычисляет следующая функция:

```
InputV asymInitLk1(SymKey lk1, PrivKey ussk, PubKey spk);
```

Семантика

Эта функция вычисляет сообщение инициализации initLk1 в соответствии с пунктом 7.3.

Остальные функции **лестницы ключей** выполняются следующим образом:

```
keyLadder(SymKey lk1, ulong cwUri, uchar acf[15], uchar ark[16],
    PubKey popk[16], SSCnfg clCnf[16], uchar XT[32], ulong spkUri, uint nSpk, PubKey spk[16],
    uchar nElk, SymKey elk[32])
```

Семантика

Эта функция вычисляет оставшуюся часть **лестницы ключей** с использованием lk1 в качестве результата блока D в **лестнице ключей** для получения результата CW для **безопасного видеотракта**

Функции **механизма аутентификации** для вычисления ключа AK выполняются следующим образом:

```
SymKey AuthMech(InputV inputV, uchar acf[15], uchar ark[16],
    PubKey pk[16], SSCnfg clCnf[16], char XT[32], ulong spkUri, uint nSpk,
    uint spkIndx, PubKey spk[16])
```

Семантика

Эта функция вычисляет **механизм аутентификации** до AK и выдает результат.

Чтобы использовать вычисленный ключ AK, с помощью интерфейса "вызов–ответ" и функционального блока d в **механизме аутентификации** согласно разделу 8 [ITU-T J.1015] определяется следующая функция:

```
uchar[16] AuthMechResponse(SymKey ak, uchar[16] challenge)
```

Семантика

Эта функция вычисляет ответ на входящий запрос с использованием AK в качестве ключа аутентификации, как определено в **механизме аутентификации**.

7.6 Определение ACF

Вход поля контроля повышенной безопасности (ACF) в **блок лестницы ключей** служит для определения основных характеристик режима работы. Значение параметра acf[0] определено в таблице 7-2.

Таблица 7-2 – ACF[0] для применения лестницы ключей

Acf[0]	Значение	Описание
AcfCw1Mode	0x11	Операция **лестницы ключей,** определенная в настоящей Рекомендации. acf [1]..acf[14] устанавливаются в 0x00
AcfAk1Mode	0x12	Операция **механизма аутентификации** определена в настоящей Рекомендации. Значение acf[1] именуется **AkModeField**. Допустимые значения определены в таблице 7-3. Acf[2]..acf[14] устанавливаются в 0x00
Зарезервировано	Другие	Зарезервированы для использования в будущем

Следующее дополнительное определение на языке C для этого режима AcfCw1Mode предназначено для применения в качестве параметра **лестницы ключей**:

```
const uchar acfCw1Mode= { AcfCw1Mode, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
```

Следующее дополнительное определение на языке C для режима AcfAk1Mode предназначено для применения в качестве параметра л**естницы ключей**:

```
const uchar acfAk1Mode= { AcfAk1Mode, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
```

Таблица 7-3 – Определение AkModeField для AcfAk1Mode

Регистр	Бит	Значение		Описание
AkUseFlag	8	0b0	**AkUseAS**	AK применяется только для **усовершенствованной системы безопасности**
		0b1	**AkUseCI**	AK применяется для **клиента ECI**
AkOnline	7	0b0	**AkOffline**	AK устанавливается в однонаправленный "автономный" режим. Вызовы/ответы могут быть рассчитаны предварительно
		0b1	**AkOnline**	AK устанавливается с использованием случайного одноразового кода AKRK, для чего потребуется вычисление ответа на запрос в режиме онлайн
AkAsAppl, только если AkUseFlag= **AkUseAS**, иначе зарезервировано	0..3	0x0	**AkConfigAuth**	Аутентифицирует элемент конфигурации **сегмента AS**
		0x1	**AkLdUssk**	Использует AK для дешифрования и загрузки ключа uSSK **микросервера**
		0x2	**AkCIImg**	Использует AK для дешифрования ключа, необходимого для дешифрования загружаемого образа **клиента ECI**
		0x3.. 0xF	Зарезервиро-вано	Зарезервировано для использования в будущем
RFU	Другое	0		Зарезервировано для использования в будущем. Значение устанавливается в ноль

8 Сегмент повышенной безопасности

8.1 Введение в тему "Сегмент повышенной безопасности"

Усовершенствованная система безопасности содержит информацию о состоянии каждого загруженного **клиента ECI**. Идентификатором, который связывает **клиента ECI** со **сегментом AS**, служит открытый ключ **системы управления платформой** (POPK) **клиента ECI**. **Хост ECI** загружает POPK **клиента ECI** в доступный **сегмент AS.** С этого момента состояние **сегмента AS** связано с этим **клиентом ECI**. В любой значимой операции **сегмента AS** в качестве входных данных используется POPK, так что результат относится только к связанному с ним **клиенту ECI** и для других не имеет смысла.

Устойчивость хоста ECI гарантирует, что **клиент ECI** может обращаться только к информации назначенного ему сегмента: в случае если **хост ECI** функционирует правильно, он гарантирует, что только назначенный **клиент ECI** сможет получить доступ к своему собственному **сегменту AS**. В случае если **хост ECI** каким-то образом взломан, то "механизм блокировки" POPK гарантирует, что значимые результаты в виде ключей дешифрования (CW) и связанных с ними **свойств контента** или ключа аутентификации (AK) смогут давать только входные данные AS, поступающие в **сегмент AS** и образующие когерентный набор.

Если **хост ECI** решит перепрофилировать **сегмент AS** для другого **клиента ECI**, то любое созданное состояние предыдущего **клиента ECI** (POPK) стирается.

8.2 Определение сегмента AS

8.2.1 Общие положения

Сегмент AS определяется в терминах ***переменных состояния*** и ***входных переменных*** для ***функций изменения состояния***. Значения представления состояния, входных и выходных данных в этом разделе выбираются таким образом, чтобы выполняемые над ними операции определялись в терминах заданных здесь двоичных представлений. В частности, это относится к их включению в криптографические вычисления и вычисления **лестницы ключей**. При фактической реализации могут быть выбраны собственные представления состояний, но для любой криптографической операции все специальные представления должны преобразовываться в указанное здесь представление входных данных.

Все переменные состояния **сегмента AS** должны быть надежно защищены от любых злонамеренных изменений. Некоторые переменные состояния содержат информацию, которая должна оставаться секретной: такие регистры должны быть надежно защищены от несанкционированного доступа. На языке C такие переменные определяются с использованием термина Secret. Любое вычисление, основанное на значении секретной переменной, должно храниться в секрете, за исключением случаев, когда такой результат явным образом используется совместно. Любое место хранения секретного значения и/или вычисление с секретным значением должны иметь ту же **устойчивость**, какая требуется для **блока лестницы ключей** [ITU-T J.1015].

Работа **сегмента AS** состоит из сеансов. Каждый сеанс выполняется в соответствии с настройками конфигурации, которые составляют часть состояния сеанса. Конфигурация сеанса устанавливается **клиентом ECI** и перед использованием должна быть аутентифицирована с помощью **механизма аутентификации** или неявных свойств аутентификации **лестницы ключей**.

Все переменные состояния и функции определены на языке C [ISO/IEC 9899]. В языке C отсутствует строгий порядок записи в том смысле, что элементы могут определяться после их применения. Массивы фиксированного размера копируются одним оператором присваивания (а не путем копирования значения указателя), как если бы они находились в "строгой" структуре.

Что касается ошибок, то определение неявной проверки ошибок делает код более читабельным. Присвоение переменной состояния или ее полю (см. определение поля) зарезервированного значения считается ошибкой. Если правая часть такого выражения присваивания основана на одном параметре функции, то для этого параметра возвращается признак ошибки: значение –i параметра i.

Все значения полей и переменных по умолчанию равны 0, если явно не указано иное.

8.2.2 Определение состояния сегмента AS

8.2.2.1 Состояние сегмента и сеанса

Состояние **сегмента AS** определяется как общее состояние сегмента и сеанса в одном сеансе шифрования или дешифрования. Структура состояния сегмента определена ниже. Поля определены в таблице 8-1.

```
#define NSLOTS       /* (максимальное) количество сегментов */
#define NSESSIONS    /* максимальное) количество сеансов */
#define MaxSpkEncr 4 /* максимальное количество значений SPK шифрования */

typedef SessionState {
    bool            active;
    uint            configAuthMode:4;
    uint            mh;
    SessionConfig   config;
    PubKey          spk;
    ulong           spkUri;
    uchar           spkIndx;
    int             coupledSessionId;
    uint            nEncr;
    PubKey          encrSpk[MaxSpkEncr];
    PubKey          encrPopk[MaxSpkEncr];
    ulong           encrCwUri;
    Secret SymKey   lk1;
    Secret PrivKey   ussk;
    RkState rkState;
    importExportState ies;
} SessionState ;

typedef struct SlotState {
    uint            version:4;
    uint            slotMode:4;
    uint            clientCheckFlag:1;
    uint            reserved:3;
    uint            POClRLVnr: 24;
    PubKey          popk;
    SymKey          slotRk;
    Secret SymKey   akClient;
    SessionState    se[NSESSIONS];
} FixedSlotState;

SlotState ss[NSLOTS]
```

Таблица 8-1 – Определение структуры состояний сегмента AS

Поле	Описание
active	Истинно, если сеанс активен, в противном случае – ложно. Значение по умолчанию – ложно
configAuthMode	Режим, с помощью которого была выполнена аутентификация конфигурации сегмента. Допустимые значения: **ConfigAuthModeNone:** 0x0, конфигурация сегмента не аутентифицирована. **ConfigAuthModeAk1:** 0x1, конфигурация сегмента аутентифицирована с использованием механизма AK, как определено в пункте 8.2.4.8. Все остальные значения зарезервированы
Mh	Дескриптор медиаконтента, с которым связан сеанс **сегмента AS**
clientCheckFlag	Загружен новый **клиент ECI**. При инициализации сеанса выполняется проверка **POClRLVnr**. Значение по умолчанию – 0b1
Reserved	Поле зарезервировано; устанавливается в 0
POClRLVnr	Номер версии **списка аннулированных сертификатов** клиента **системы управления платформой**, которая использовалась для проверки **клиента ECI** перед загрузкой. Проверяется при инициализации каждого сеанса **клиента ECI** по наименьшему номеру версии, ожидаемой **клиентом ECI**
slotConfig	Конфигурация сегмента
spk	Открытый ключ, используемый для вычисления LK1 и AK
spkUri	Информационный регистр правил применения векторов SPK, использовавшийся для вычисления LK1
spkIndx	Местоположение регистра выбора индекса SPK в векторе SPK для вычисления LK1
coupledSessionId	Применяется, только если сеанс **сегмента AS** находится в режиме дешифрования. Второй сеанс (дешифрование) спарен с данным. Декодированные потоки контента объединяются, а **свойства контента** сравниваются. Значение по умолчанию: –1
nEncr	Количество входных значений SPK/POPK, использовавшихся для шифрования (исключая spk сегмента)
encrSpk	Значения SPK для шифрования контента с помощью **лестницы ключей**
encrPopk	Значения POPK для шифрования контента с помощью **лестницы ключей**
encrCwUri	Значения CwUri для шифрования контента с помощью **лестницы ключей**
lk1	Ключ связи верхнего уровня для вычисления управляющих слов с помощью **лестницы ключей**
Ussk	Секретный ключ **микросервера (**для применений **микросервера)**

Таблица 8-1 – Определение структуры состояний сегмента AS

Поле	Описание
rkState	Состояние случайного ключа сеанса
ies	Состояние импорта/экспорта сеанса
version	Версия состояния сегмента. Допустимые значения: 0x1: версия 1. Все остальные значения зарезервированы
slotMode	Режим работы сегмента. Допустимые значения: **SlotModeDecr:** 0x1, сегмент работает в режиме дешифрования. **SlotModeEncr:** 0x2, сегмент работает в режиме шифрования. Все остальные значения зарезервированы
popk	Открытый ключ **клиента ECI**, использующего данный сегмент
slotRK	Случайное число, используемое в протоколах ответа на онлайн-запрос, например **сервера подготовки**. Значение устанавливается при инициализации сегмента
akClient	Ключ аутентификации для целей обработки клиента
se	Состояние сеанса (для всех сеансов **сегмента AS**)
ss	Состояние **сегмента AS** (для всех сегментов)

Если не указано иное, то значение по умолчанию для каждого элемента состояния при инициализации равно нулю.

8.2.2.2 Конфигурация дешифрования

Соответствующее состояние конфигурации сеанса **сегмента AS** в режиме дешифрования определено в приведенном ниже коде на языке C, а его описание дано в таблице 8-2. В ней определены детали работы сеанса сегмента AS в режиме дешифрования. Эти данные можно аутентифицировать, выполняя соответствующие вычисления **механизма аутентификации** или **лестницы ключей**.

```
typedef struct DecryptConfig {
    uint            configVersion:4;
    uint            reserved1:4;
    uint            klModeAuth:1;
    uint            akModeAuth:1;
    uint            rkKlMode:1;
    uint            spk0NoDecrypt:1;
    uint            reserved2:6;
    RKMode          rkDecrMode;
    EciRootState    minEciRootState;
    uint            minClientVersion:24;
} DecryptConfig;
```

Таблица 8-2 – Определение структуры DecryptConfig

Поле	Описание
configVersion	Версия конфигурации дешифрования. Значение определено как 0x1: версия 1. Все остальные значения зарезервированы. Если в этом поле не содержится допустимое значение, то в сеансе **сегмента AS** должно быть отказано в выполнении каких-либо функций перехода состояния
reserved1	Зарезервированное поле; устанавливается в 0
klModeAuth	Если установлен этот бит, то при всех вычислениях **лестницы ключей** в сеансе **сегмента AS** для аутентификации применяется ClientConfig. Сам этот бит аутентифицируется при всех вычислениях **лестницы ключей**
akModeAuth	Если установлен этот бит, то, прежде чем разрешить любой расчет **лестницы ключей**, в сеансе **сегмента AS** проверяется, установлено ли для configAuthMode значение ConfigAuthModeAk1. Сам этот бит аутентифицируется при всех вычислениях **лестницы ключей**
rkKlMode	Если установлен этот флаг, то во всех вычислениях **лестницы ключей** для **сегмента AS** применяется **slotRK**
spk0NoDecrypt	Если установлен этот бит, запрещается использовать spk [0] (spkIndx == 0 как входной параметр функции **лестницы ключей**) для аутентификации сегмента LK1 в режиме дешифрования
reserved2	Зарезервированное поле; устанавливается в 0
rkDecrMode	Определяет применение случайного ключа сеанса для вычислений **лестницы ключей**. См. пункт 8.2.2.5
minEciRootState	Минимальное значение для корневой версии **ECI** и корневой версии **списка аннулированных сертификатов**. Если CPS применяет **корневой ключ ECI** или корневой **список аннулированных сертификатов** для целей аутентификации **ECI** со значением, меньшим указанного в этой структуре, то в данном сеансе вычисление **лестницы ключей** не разрешается
minClientVersion	Версия **клиента ECI**. Используется для проверки номеров версий **списка аннулированных сертификатов** POPK

8.2.2.3 Конфигурация шифрования

Соответствующее состояние конфигурации сеанса **сегмента AS** в режиме шифрования определено в приведенном ниже коде C, а его описание дано в таблице 8-3.

```
typedef struct EncryptConfig {
    uint            configVersion:4;
    uint            reserved1:4;
    uint            microServerVersion:24;
    uint            asymKlMode:1;
    unit            rkKlMode:1;
    uint            reserved2:22;
    RkMode          rkEncrMode;
    uchar           basicUriTrfr;
    uint            contPropControl;
    ContProp        defaultCP;
    EciRootState    minEciRootState;
} EncryptConfig;
```

Таблица 8-3 – Определение структуры EncryptConfig

Поле	Описание
configVersion	Версия конфигурации шифрования. Значение определено как 0x1: версия 1. Все остальные значения зарезервированы. Если в этом поле не содержится допустимое значение, то в сеансе **сегмента AS** не выполняются никакие функции перехода состояния
Reserved1	Зарезервированное поле; устанавливается в 0
microServerVersion	Номер версии конфигурации **микросервера.** Также используется как минимальный номер версии **списка аннулированных сертификатов** для аутентификации **микроклиента** в режиме асимметричного **микросервера**
asymKlMode	Если установлен этот флаг, то **лестница ключей** должна работать в соответствии с асимметричным **механизмом аутентификации** клиента, определенным в пункте 7.3
rkKlMode	Если установлен этот флаг, то во всех вычислениях **лестницы ключей** применяется **slotRK**
Reserved2	Зарезервированное поле; устанавливается в 0
rkEncrMode	Определяет применение случайного ключа для вычислений **лестницы ключей**. См. пункт 8.2.2.5
basicUriTrfr	Определяет преобразования состояния базового URI из соединения импорта перед применением базового URI в качестве свойства зашифрованного контента. Значения см. в таблице 8-5
contPropControl	Определяет способ вычисления **свойств контента** зашифрованного контента. См. таблицу 8-4
defaultCP	Значение по умолчанию для всех полей свойств контента. Применением при вычислении **лестницы ключей** управляет поле contPropControl
minEciRootState	Минимальное значение для корневой версии **ECI** и корневой версии **списка аннулированных сертификатов**. Если CPS применяет **корневой ключ ECI** или корневой **список аннулированных сертификатов** для целей аутентификации **ECI** со значением, меньшим указанного в этой структуре, то вычисление **лестницы ключей** не разрешается

Поле contPropControlFields представляет собой массив из 16 двухбитовых полей. Двухбитовые поля указывают на способ управления полем Field1 "**Свойства контента**" зашифрованных выходных данных. Описание приведено в таблице 8-4. Бит 2n и бит 2n+1 CpControlFlag должны соответствовать Field1-байт n.

Таблица 8-4 – Определение CpCtrl

Имя флага	Значение	Описание
CpCtrlCopy	0b00	Байт свойств контента CP Field1 копируется из соединения импорта
CpCtrlDef	0b01	Значение байта свойств контента CP Field1 устанавливается равным значению соответствующего байта defaultCP
CpCtrlMS	0b10	Значение байта свойств контента CP Field1 задается **микросервером**
Зарезервиро-вано	0b11	Значение зарезервировано

Поле basicUriTrfr изменяет описанное выше поведение флагов CPControlFlag по отношению к полю BasicUri в случае нахождения его флага CPControlFlag в состоянии **CPCopy**. Альтернативное поведение определено в таблице 8-5.

Таблица 8-5 – Значения и описание BasicUriTrfr

Имя флага	Значение	Описание
JustCopy	0x00	Байт свойств контента Field1 копируется из соединения импорта
NoMoreCopy	0x01	Состояние basicURI свойства **RedistributionProtected** преобразуется в состояние **ViewOnly**
Зарезервиро-вано	Другое	Зарезервировано для использования в будущем
ПРИМЕЧАНИЕ. – Установив флаг BasicUriTrfr в состояние NoMoreCopy, система **микроклиента** разрешает направлять в микросервер только поток любого защищенного контента.		

8.2.2.4 Управление случайным ключом сеанса

Структура RKMode, определенная в нижеследующем коде на языке C и в таблице 8-6, определяет режим применения случайного ключа сеанса в **лестнице ключей**.

```
typedef struct RKMode {
    uint    mode:2;
    uint    limit:6;
} DecryptConfig;
```

Таблица 8-6 – Структура случайного ключа сеанса дешифрования и шифрования

Поле	Описание
mode	Определяет режим применения случайного ключа сеанса. Принимаемые значения: • 0b00: **RKModeNone**, случайный ключ сеанса не введен • 0b10: **RKModeDataLimit**, случайный ключ сеанса применяется с ограничением данных • 0x11: **RKModeTimeLimit**, случайный ключ сеанса применяется с ограничением времени 0b01: значение зарезервировано
limit	Значение определяет применяемый предел в секундах реального времени или килобайтах данных, дешифрованный или зашифрованный с момента инициализации случайного ключа. Функция limitValue() определяет применяемое фактическое предельное значение. Значение 63 зарезервировано

```
uint limitValue(uint limit) {
    uint val;

    if (limit==0) return 1;
    limit -=1;
    if (limit&0b1 == 0b0) val=2 else val=3;
    return val * (1<<(limit>>1));
}
```

8.2.2.5 Полная конфигурация сеанса

Полная информация управления конфигурацией сеанса **сегмента AS** в режиме шифрования или дешифрования содержится в определенной ниже структуре SessionConfig. В случае если **сегмент AS** находится в режиме шифрования, она включает информацию о конфигурации для последующего дешифрования.

```
typedef struct SessionConfig {
    EncryptConfig   encryptConfig; /* конфигурация для дешифрования */
    DecryptConfig   decryptConfig; /* конфигурация для дешифрования */
} SessionConfig;
```

Структура cpsEciRootState, определяющая корневое состояние **ECI** для проверки **цепочек сертификатов ECI**, определяется в приведенном ниже коде на языке C и в таблице 8-7.

```
typedef struct EciRootState {
    uchar       rootVersion;
    uint            rlVersion:24;
} EciRootState;

EciRootState cpsEciRootState; /* содержит минимальное значение из CPS */
```

Таблица 8-7 – Описание поля структуры EciRootState

Поле	Описание
rootVersion	Версия **корневого сертификата ECI**
rlVersion	Версия **списка аннулированных сертификатов**, применяемая с **корневым сертификатом**
ПРИМЕЧАНИЕ. – EciRootState обычно применяется как нижняя граница (минимальное значение) для корневой версии **ECI** и версии **списка аннулированных сертификатов** при загрузке информации **хоста ECI** и **клиента ECI**.	

Следующая определяемая функция проверяет, достаточно ли cpsEciRootState для перехода к вычислению ключа:

```
bool cpsEciRootStateOk(uint slotId, uint sessionId) {
  if (ss[slotId].slotMode == SlotModeDecr)
    return
        (cpsEciRootState.rootVersion >=
         ss[slotId].se[sessionId].config.decryptConfig.minEciRootState.rootVersion)
     && (cpsEciRootState.rlVersion >=
         ss[slotId].se[sessionId].config.decryptConfig.minEciRootState.rlVersion);

  if (ss[slotId].slotMode == SlotModeEncr)
    return
        (cpsEciRootState.rootVersion >=
         ss[slotId].se[sessionId].config.encryptConfig.minEciRootState.rootVersion)
     && (cpsEciRootState.rlVersion >=
         ss[slotId].se[sessionId].config.encryptConfig.minEciRootState.rlVersion);

  /* следующее происходить не должно */
  return false;
}
```

Предварительные условия

- Инициализирован slotId **сегмента AS**.

8.2.2.6 Состояние случайного ключа сеанса

Для каждого сеанса дешифрования или шифрования, связанного со **сегментом AS**, **сегмент AS** хранит информацию о состоянии случайного ключа, как определено в нижеследующем коде на языке C и описано в таблице 8-8.

```
typedef struct RkState {
    SymKey  rkCurrent;
    SymKey  rkNext;
    ulong   limitCounter;
} RkState;
```

Таблица 8-8 – Описание поля состояния случайного ключа RkState

Поле	Описание
rkCurrent	Текущий случайный ключ, используемый для ввода в **лестницу ключей** при вычислении CW
rkNext	Следующее значение случайного ключа, который вводится в **лестницу ключей** для вычисления CW
limitCounter	Счетчик, показывающий состояние использования текущего ключа в единицах, связанных с предельным значением, применяемым к ключу. Значение соответствует числу оставшихся единиц, которые еще можно зашифровать или дешифровать, в зависимости от значения CW, вычисленного с помощью ключа rkCurrent

Значение поля **limitCounter** получает приращение при каждом применении CW.

ПРИМЕЧАНИЕ. –В случаях конкретной реализации счетчик может быть эффективно использован в составе безопасного видеотракта.

8.2.2.7 Состояние импорта/экспорта

С каждым сеансом шифрования связан один сеанс дешифрования, из которого он импортирует перешифровываемый контент. Импорт должен быть возможен одновременно для двух (или более) групп сеансов экспорта, что позволяет осуществлять плавное переключение.

Два сеанса дешифрования могут быть спарены. Это позволяет объединить разные подпотоки, которым требуются разные управляющие слова (рассчитанные одним и тем же **сегментом AS**) в один составной поток с единым набором **свойств контента** перед отправкой на стандартные выходы или перед экспортом. В процессе объединения **система AS** проверяет эквивалентность **свойств контента** объединенных потоков.

ПРИМЕЧАНИЕ. – Сравнение **свойств контента** также может охватывать идентификатор группы экспорта, что гарантирует, что обработка цепочки экспорта, требуемая для обоих спаренных сеансов, эквивалентна.

В сочетании с состоянием случайного ключа это состояние связки сеансов в **сегменте AS**. Состояние сеанса "se" определено в нижеследующем коде на языке C; описания полей приведены в таблице 8-9.

```
#define MaxExpGroupIds 2

typedef struct ImportExportState {
    int     importSlotId;
    int     importSession;
    uchar   expGrpId[MaxExpGrpId];
    bool    importPermitted[MaxExpGrpId];
    RkState rkState;
} ImportExportState;

#define ImportNone -1
```

Таблица 8-9 – Определение структуры ImportExportState

Поле	Описание
importSlotId	Применимо, только если **сегмент AS** находится в режиме шифрования. Значением является номер сегмента, из которого импортируется контент ("сегмент импорта"). Значение по умолчанию: –1
importSession	Применимо, только если **сегмент AS** находится в режиме шифрования. Значением является номер сеанса в сегменте импорта, из которого импортируется контент. Значение по умолчанию: –1
expGrpId[eid]	Применимо, только если **сегмент AS** находится в режиме шифрования. Идентификатор группы экспорта экспортирующего **сегмента AS**, из которого можно импортировать контент. Значение 0x00 зарезервировано
importPermitted[eid]	Применимо, только если **сегмент AS** находится в режиме шифрования. Значение "истинно" устанавливается, если **expGrpId [eid]** разрешено экспортирующим **сегментом AS**; в противном случае устанавливается значение "ложно". Значение по умолчанию: ложно
rkState	Состояние случайного ключа сеанса для данного сеанса

Если сеанс дешифрования сброшен или инициализирован повторно, **система AS** удаляет соответствующий сеанс импорта (устанавливает значение "ложно" в соответствующем поле ImportPermitted). При перезагрузке или повторной инициализации **сегмента AS** **система AS** сбрасывает все сеансы **сегмента AS**.

8.2.3 Аутентификация свойств контента

Клиенты ECI, выполняющие функции дешифрования, предоставляют **хосту ECI** значения свойств контента через соответствующий API свойств контента. **Хост ECI** вводит эти значения в **усовершенствованную систему безопасности** в сочетании с данными, необходимыми для вычисления управляющего слова для соответствующего контента. **Усовершенствованная система безопасности** обеспечивает надлежащее применение **свойств контента** и проверяет **свойства контента**, используя их для вычисления входных данных **блока лестницы ключей AS** на языке C.

Микросерверы используют **свойства контента**, переданные и/или обработанные **системой AS** или **клиентом ECI**, применяя описанный выше механизм для вычислений в процессе вычисления входных данных **лестницы ключей** на языке C. **Сегменты AS**, используемые в режиме шифрования, сравнивают **свойства контента**, предоставленные **микросервером**, с теми, которые были переданы ресурсом дешифрования в соответствии с параметрами конфигурации **микросервера**. При обнаружении несоответствия шифрование прекращается.

Для целей аутентификации и проверки **свойства контента** объединяются в последовательность байтов в два этапа. На первом этапе меньшие поля свойств контента фиксированной длины объединяются в поле *field1*. Байт fieldControl проверяет наличие полей свойств контента размером в байтах для аутентификации. На втором этапе более длинные поля свойств контента объединяются в последовательность байтов *field2*. *Field1 u field2* объединяются и вводятся в хеш-функцию, которая

собирает все поля в 128-битовое значение для ввода в **лестницу ключей** на языке C. Структура поля field1 представлена в таблице 8-10.

Таблица 8-10 – Определение структуры field1

Имя	Тип	Номер байта	Описание
fieldControl	FieldControl	0,1	Это поле определяет 16-битовое значение с младшими значащими битами в байте 0. См. таблицу 8-11
basicUri	byte	2	Значение этого поля должно соответствовать спецификации типа BasicUri, [ITU-T J.1012], таблица 9.8.2.5.1-1
outputControl	byte [2]	3–4	Значение этого поля должно соответствовать спецификации Output Control Vector, [ITU-T J.1012], таблица 9.8.2.6.1-1
standardUri	byte [3]	5–7	Значение этого поля должно соответствовать спецификации Стандарта URI, [ITU-T J.1012], таблица 9.8.2.3.1-1
exportGroup	byte	8	Интерпретируется как целое число без знака, представляющее идентификатор группы экспорта, применяемой к контенту. Значение 0 интерпретируется **хостом ECI** как запрет экспорта; значения 0x80–0xFF зарезервированы
parentalAuth	byte	9	Соответствует параметру ParCond.basicCondition, определенному в [ITU-T J.1012], пункт 9.8.2.8.1-1, биты [0..5] которого установлены в 0b000000
Reserved	byte [6]	10–15	**Хосты ECI**, соответствующие настоящей Рекомендации, устанавливают эти байты в 0x00

Таблица 8-11 – Определение структуры FieldControl

Имя	Бит(ы)	Описание
bit-<n>	2–16	Этот бит управляет проверкой байта <n> поля field1. Значение 0b1 указывает на то, что значение байта <n> проверяется и должно совпадать с указанным полем, значение 0b0 указывает на то, что байт <n> не проверяется, а в поле field1 вместо байта <n> должно использоваться значение 0x00. Бит 2 устанавливается в 0b1 при его использовании в качестве входного значения для вычисления управляющего слова дешифрования. Это гарантирует, что basicUri всегда аутентифицирован по значению, используемому на момент шифрования контента
Field2ctrl	0–1	Значение 0b00 указывает на отсутствие поля field2. Значение 0b01 указывает на то, что поле присутствует и используется кодировка, определенная ниже. Значения 0b10 и 0b11 зарезервированы и не используются

В определении поля Field2 используется структура "признак/длина/значение", причем общее поле длины обеспечивает общую целостность. Структура поля Field2 определена ниже в этом разделе.

Функция computeField1Decrypt определяет логику выбора следующего шага аутентификации:

```
    void computeField1Decrypt(uchar field1[16], uchar result1[16]) {
        int i;
        ushort fieldControl = field1[0] + field1[1]<<8;

        result1[0] = field1[0];
        result1[1] = field1[1];
        for (i=2; i<16; i++)
            if (fieldControl>>i & 0b1)
                result1[i] = field[i];
            else
                result1[i] = 0x00;
    }
```

Сегмент AS в режиме шифрования вычисляет входные данные для аутентификации свойств контента, обозначенные как result1, а также поле cpMask, которое служит для сравнения байтов field1 с полем field1 контента сеанса импорта:

```
    void computeField1Encrypt(
                uchar msField1[16],/* поле field1 CP от клиента микросервера */
result1[16],            /* результат CP для аутентификации при вычислении CW */
                ushort cpMask,              /* маска результата для сравнения msField1 с версией
                                                поля CP field1 клиента */
                EncryptConfig ssEncrypt /* конфигурация шифрования сегмента AS */
    ) {
        int i;
        uchar cp[16]; /* вычисляемое значение CP */

        /* установка управляющих байтов свойств контента */
        cp[0] = ssEncrypt.defaultCp[0];
```

```
        cp[1] = ssEncrypt.defaultCp[1];
        mask = 0x0000;

        /* обработка правил contPropControl для вычисления cp p */
        for (i=2; i<16; i++) {
            switch (ssEncrypt.contPropControl>>(2*i) && 0b11) {
                case CpCtrlCopy: /* копируется из импортирующего клиента */
                    if (i==2) { /* байт базового URI */
                        /* process basicUriTrfr */
                        switch (ssEncrypt.basicUriTrfr) {
                        case BasicUriTrfrNoChange:
                            cp[i]= msField1[i];
                            break;
                        case BasicUriTrfrNoMoreCopy:
                            if ((clField1[2]&0b11) == RedistributionProtected)
                                cp[i]= (msField1[1] & 0xFC) + ViewOnly;
                            else
                                cp[i]= msField1[i];
                            break;
                    } else {   /* все остальные байты CP */
                        cp[i]= msField1[i];
                    }
                    cpMask += 1<<i; /* сравнивается с импортирующим клиентом */
.                   break ;
                case CpCtrlDef: /* устанавливается в значение CP по умолчанию согласно конфигурации
                                 *?
                    cp[i] = ssEncrypt.defaultCP[i] ;
                    break ;
                case CpCtrlMS: /* определяется программным обеспечением микроклиента пользователя */
                    cp[i] = msField1[i];
                    break;
            }
        }

        /* вычисляет входные данные для функции аутентификации так же, как и для дешифрования */
        computeField1Decrypt(cp, result1);
    }
```

Поле Field2 представляет собой структурированную последовательность байтов, как указано ниже:

```
typdef struct Field2 {
    uint length;        /* количество байтов контента, должно быть кратным 4 */
    byte content[]; /* контент определен ниже */
} Field2;
```

Поле контента структуры Field2 содержит последовательность структур LargeProperty, каждая из которых имеет уникальный признак. LargeProperty определяется следующим кодом на языке C:

```
typdef struct LargeProperty {
    uint  propertyTag;  /* см. таблицу 8.2-12 */
    uint  length;       /* длина поля свойств в байтах
    byte  property[];   /* содержит фактическое значение свойства */
    byte  padding[];    /* дополнительные байты установлены в 0x00, чтобы сделать LargeProperty
                           целым 4-байтовым числом /
} LargeProperty;
```

Значения поля propertyTag структуры largeProperty и определения соответствующих свойств приведены в таблице 8-12.

Таблица 8.12 – Значения и смысл поля признака largeProperty

Значение propertyTag	Свойство
0x00000000	Зарезервировано
0x00000001	Соответствует параметру **data** сообщения **setDcrMarkBasic**, определенного в [ITU-T J.1012], пункт 9.8.2.7.5
0x00000002	Соответствует параметру **data** сообщения **setDcrMarkExt**, определенного в [ITU-T J.1012], пункт 9.8.2.7.6
0x00000003	Соответствует параметру **custURI** сообщения **setDcrCustUri**, определенного в [ITU-T J.1012], пункт 9.8.2.4.1
Другое	Зарезервировано для использования в будущем

Система AS может отказаться от любых данных, превышающих ее пропускную способность для поля *field2*.

Система AS проверяет согласованность любых параметров данных Field2, осуществляя следующие проверки:

- длина структур, составляющих LargeProperty, равна полю длины структуры Field2;
- байты заполнения всех структур, составляющих LargeProperty, равны 0x00.

Соответствующие входные данные C для **лестницы ключей** вычисляются по значениям result1 и field2 с использованием следующего кода на языке C:

```
void computeInputC(uchar result1[16], unchar *field2, uchar input_C[16])
{
    uchar hash2[16], hashIn[32];
    uint i, length;

    if (result1[0] & 0b11 == 0x00) {
        /* field2 не включается */
        for (i=0; i<16; i++) hashIn[i] = result1[i];
        asHash(hashIn, 16, 128, input_C);
    } else if result[0] & 0b11 = 0x01) {
        /* field2, включаемое для input-C  */
        length = (Field2 *)field2->length + 4;
        asHash(field2, length, 256, hash2);
        for (i=0; i<16; i++) hashIn[i] = result1[i];
        for (i=0; i<32; i++) hashIn[16+i] = hash2[i];
        asHash(hashIn, 48, 128, input_C);
    }
}
```

asHash – это определенная в разделе A.1 хеш-функция байтовой последовательности в первом параметре, длины байтовой последовательности во втором параметре, битовой длины результата в третьем параметре и результата в последнем параметре.

Устойчивость вычисления внешнего хеша (прямое вычисление) должна быть как минимум такой же высокой, что и при вычислении внутреннего хеша. Степень **устойчивости** хеша отражает трудоемкость создания расхождения между любыми входными данными хеш-функции и их применением в качестве свойства контента, а также манипулирования хеш-функцией и/или ее результатом.

Примером разных уровней **устойчивости** двух вычислений хеша является то, что внешний хеш может быть рассчитан устойчивым аппаратным блоком, а внутренний – с помощью устойчивой реализации программного обеспечения.

8.2.4 Функции сегмента AS

8.2.4.1 Обзор

Система AS может выполнять различные функции от имени **клиентов ECI**, действуя через **хост ECI**. Эти функции составляют основу API повышенной безопасности в соответствии с [ITU-T J.1012]. "Событие" сообщает **клиенту ECI** об асинхронно происходящем событии. Никакой ответ невозможен. Все остальные функции представляют собой асинхронные или синхронные сообщения, инициированные **клиентом ECI**; их возвращаемые значения указывают статус ответа. Эти функции перечислены в таблице 8-13.

Таблица 8.13 – Обзор функций повышенной безопасности

Имя функции	Описание	Пункт
reqAsInitSlot	Инициализация **сегмента AS**	8.2.4.2
reqAsAStartDecryptSession	Запуск сеанса дешифрования в **сегменте AS**	8.2.4.3
reqAsCoupleDecryptSession	Спаривание двух сеансов дешифрования в один	8.2.4.3
reqAsDecoupleDecryptSession	Разделение двух спаренных сеансов дешифрования	8.2.4.3
reqAsStartEncryptSession	Запуск сеанса шифрования	8.2.4.3
callAsNextKeySession	Переход к следующему случайному ключу	8.2.4.3
reqAsStopSession	Остановка сеанса	8.2.4.3
reqAsExportConnSetup	Установление **соединения экспорта** между сеансами дешифрования и шифрования	8.2.4.4
reqAsExportConnEnd	Завершение текущего сеанса экспорта	8.2.4.4
reqAsLoadLk1	Загрузка ключа связи верхнего уровня в **лестницу ключей** для данного сеанса	8.2.4.5

Таблица 8.13 – Обзор функций повышенной безопасности

Имя функции	Описание	Пункт
reqAsComputeEncrCw	Вычисление управляющего слова шифрования	8.2.4.6
reqAsComputeDecrCw	Вычисление управляющего слова дешифрования	8.2.4.7
reqAsComputeAkClient	Вычисление ключа аутентификации для использования **клиентом ECI**	8.2.4.8
reqAsClientChalResp	Использование ключа аутентификации от имени **клиента ECI**	8.2.4.8
reqAsAuthDecrConfig	Аутентификация конфигурации сеанса с помощью **механизмов аутентификации** (режим дешифрования)	8.2.4.9
reqAsAuthEncrConfig	Аутентификация конфигурации сеанса и параметров шифрования с помощью **механизмов аутентификации** (режим шифрования)	8.2.4.9
reqAsLdUssk	Загрузка секретного ключа **микросервера**	8.2.4.10
reqAsMInikLk1	Вычисление асимметричного сообщения инициализации **микроклиента**	8.2.4.11
reqAsClientImageDecrKey	Вычисление ключа дешифрования образа **клиента ECI**	8.2.4.12
getAsSlotRk	Считывание случайного ключа сегмента	8.2.4.13
getAsSessionRk	Считывание случайного ключа сеанса	8.2.4.13
getAsSessionLimitCounter	Считывание оставшихся блоков случайного ключа сеанса	8.2.4.13
setAsSessionLimitEvent	Отправка сообщения о событии при достижении предельного значения оставшихся блоков	8.2.4.13
reqAsEventSessionLimit	Сообщение о событии при достижении предельного значения	8.2.4.13
getAsClientRnd	Получение нового случайного числа для приложений **клиента ECI**	8.2.4.13
getAsSC	Получение текущего состояния поля управления скремблированием для контента в ходе сеанса	9.9
reqAsEventCpChange	Сообщение о событии при изменении свойств контента в импортируемом контенте в ходе сеанса шифрования	9.9
setAsPermitCPChange	Разрешение/запрет изменений CP свойств импортируемого контента, влияющих на выбор управляющего слова для шифрования в сеансе шифрования	9.9
setAsSC	Установка поля управления скремблированием зашифрованного контента в ходе сеанса шифрования	9.9
reqAsEventSC	Сообщение о событии при изменении поля управления скремблированием в ходе сеанса	9.9

Псевдокод, приведенный в подпунктах этого пункта, содержит коды ошибок в качестве возвращаемых значений функций. Значения кода ошибки вместе со словесным описанием приведены в пункте 8.2.4.15.

8.2.4.2 Инициализация сегмента AS

Во время загрузки **хост ECI** резервирует **сегмент AS** в **усовершенствованной системе безопасности** от имени каждого загружаемого **клиента ECI**. **Хост ECI** вызывает функцию reqAsInitSlot, как определено ниже. Вся информация о состоянии **сегмента AS** приводится в состояние по умолчанию; любое **соединение экспорта** сбрасывается. **Хост ECI** загружает **клиента ECI** с использованием ядра загрузчика (см. раздел 11). Параметр *POClRLVnr* **сегмента AS** отражает минимальный номер версии **списка аннулированных сертификатов** POC, используемой для проверки образа клиента. Это значение проверяется при инициации сеанса **клиентом ECI**.

```
int  reqAsInitSlot(uint slotId, ECI_Certificate_Chain popkChain,
    uint slotVersion, slotMode)
```

Семантика

Вся информация о состоянии **slotId сегмента AS** приводится в состояние по умолчанию; любое **соединение экспорта** сбрасывается.

Для загрузки POPK требуется цепочка обработки для ввода в систему CPS. Правила обработки цепочек POPK определены в пункте 10.4. Цепочка ECI_Certificate_Chain определена в пункте 5.4.1 [ITU-T J.1012] . После успешной проверки выполняется следующий код на языке C:

```
/* инициализация состояния сегмента */
ss[slotId].popk = /* проверенное значение popk, возвращаемого CPS */;
ss[slotId].POClRLVnr = /* значение, используемое для проверки образа клиента */;
ss[slotId].version = slotVersion;
ss[slotId].slotMode = slotMode;
ss[slotId].configAuthMode = ConfigAuthModeNone;
ss[slotId].rkSlot = rnd128();
return ErrOk;
```

Функция rnd128() возвращает случайное 128-битовое число, как указано в разделе A.3, в виде массива из 16 универсальных строк символов.

8.2.4.3 Управление сеансом сегмента AS и случайным ключом

Сегмент AS поддерживает разные состояния сеанса для разных одновременных сеансов. Сеансы сегмента запускаются и останавливаются следующими функциями:

```
int reqAsAStartDecryptSession(uint slotId, ushort mh, PubKey spk,
        SessionConfig config, uint *sessionId)
```

Семантика

Выполняется следующий код на языке C:

```
if (ss[slotId].slotMode != slotModeDecr) return ErrSlotMode;

/* проверка использования действительного списка аннулированных сертификатов клиента */
if (config.decryptConfig.clientVersion >
    ss[slotId].clientPOClRLVnr) return ErrRevocEnforce;

/* найти любой свободный sessionId; годится любой алгоритм */
int i=0;
while (i<NSESSIONS && ss[slotId].se[i].active) i++;
if (i==NSESSIONS) return ErrNoMoreSessions;
/* i содержит блок администрирования неактивного сеанса */
*sessionId = i;

/* инициализация состояния сеанса */
ss[slotId].se[i].active = true;
ss[slotId].se[i].mh = mh;
ss[slotId].se[i].coupledSessionId = -1;
ss[slotId].se[i].importPermitted = false;
ss[slotId].se[i].spk = spk;
ss[slotId].se[i].config = config;
ss[slotId].se[i].rkState.rkCurrent = rnd128();
ss[slotId].se[i].rkState.rkNext      = rnd128();
ss[slotId].se[i].rkState.limitCounter =
            limitValue(config.decryptConfig.rkDecrMode.limit);

if (!cpsEciRootStateOk(sdlotId,i)){
    ss[slotId].se[i].active = false;
    return ErrRevocEnforce;
}

return ErrOk;
```

Предварительные условия

- **Сегмент AS** успешно инициализирован.

ПРИМЕЧАНИЕ. – Параметр mh (дескриптор медиаконтента) позволяет **хосту ECI** идентифицировать сеанс дешифрования AS, связанный с началом сеанса дешифрования контента. Он не используется самой **системой AS**.

Для спаривания двух инициализированных сеансов предусмотрена функция coupleDecrypSessions. Второй сеанс спаривания с первым; первый становится основным дескриптором комбинированного контента.

```
int reqAsCoupleDecryptSession(uint slotId, uint sId1, uint sId2)
```

Семантика

Выполняется следующий код на языке C:

```
if (ss[slotId].slotMode != slotModeDecr) return ErrSlotMode;
if (!ss[slotId].se[sId1].active) return ErrParam2;
if (!ss[slotId].se[sId2].active) return ErrParam3;
if (ss[slotId].se[sId1].coupledSessionId != -1) return ErrSession1Coupled;
if (ss[slotId].se[sId2].coupledSessionId != -1) return ErrSession2Coupled;

se[slotId][sId1] = sId2;
/* информирование безопасного видеотракта о спаривании сеансов */

return ErrOk;
```

Предварительные условия

- Оба сеанса **сегмента AS** инициализированы успешно.

Для разделения спаренных сеансов может быть вызвана следующая функция:

```
int reqAsDecoupleDecryptSession(uint slotId, uint sessionId)
```

Семантика

Выполняется следующий код на языке C:

```
if (ss[slotId].slotMode != slotModeDecr) return ErrSlotMode;
if (!ss[slotId].se[sessionId].active) return ErrParam2;
if (se[slotId][sessionId].coupledSessionId == -1)
    return ErrSessionNotCoupled;

ss[slotId].se[sessionId].ies.coupledSessionId = -1;
/* информирование безопасного видеотракта о распаривании сеансов */

return ErrOk;
```

Предварительные условия

- Сессии предварительно спарены.

Функция инициирования сеанса шифрования:

```
int reqAsStartEncryptSession(uint slotId, ushort mh, uint importSlotId,
        int importSessionId, PubKey spk, SessionConfig config,
        uint nEncr, PubKey encrSpk[MaxSpkEncr],
        PubKey encrPopk[MaxSpkEncr], ulong encrCwUri, uint *sessionId)
```

Семантика

Выполняется следующий код на языке C:

```
If (ss[slotId].slotMode != slotModeEncr) return ErrSlotMode;
if (0 > nEncr || nEncr >= MaxEncr) return ErrParam4;

/* найти свободный sessionId; годится любой алгоритм */
int i=0;
while (i<NSESSIONS && ss[slotId].se[i].active) i++;
if (i==NSESSIONS) return ErrNoMoreSessions;
/* i содержит блок администрирования неактивного сеанса */

/* проверка использования действительного списка аннулированных сертификатов клиента */
if (config.encryptConfig.microServerVersion >
    ss[slotId].clientPOClRLVnr) return ErrRevocEnforce;

*sessionId = i;

/* инициализация информации о состоянии сеанса */
ss[slotId].se[i].active=true;
ss[slotId].se[i].mh = mh;
ss[slotId].se[i].spk = spk;
ss[slotId].se[i].config = config;
ss[slotId].se[i].encrCwUri = encrCwUri;

int j;
for (j=0; j<nEncr; j++) {
  ss[slotId].se[i].encrSpk[j] = encrSpk[j];
  ss[slotId].se[i].encrPopk[j] = encrPopk[j];
}

/* инициализация состояния случайного ключа */
ss[slotId].se[i].rkState.rkCurrent = rnd128();
ss[slotId].se[i].rkState.rkNext    = rnd128();
ss[slotId].se[i].rkState.limitCounter =
            limitValue(config.encryptConfig.rkEncrMode.limit);

/* инициализация состояния импорта */
ss[slotId].se[i].importSlotId = importSlotId;
ss[slotId].se[i].importSession = importSessionId;

if (!cpsEciRootStateOk(sdlotId,i)){
    ss[slotId].se[i].active = false;
    return ErrRevocEnforce;
}

return i;
```

Предварительные условия

- **Сегмент AS** успешно инициализирован.

Хост ECI может изменять состояние случайного ключа сеанса (делая следующее состояние текущим), используя следующую функцию:

```
int callAsNextKeySession(uint slotId, uint sessionId)
```

Семантика

Выполняется следующий код на языке C:

```
if (!ss[slotId].se[sessionId].active) return ErrNoSuchSession;

ss[slotId].se[sessionId].rkCurrent = ss[slotId].se[sessionId].rkNext;
ss[slotId].se[sessionId].rkNext = rnd128();
if (ss[slotId].slotMode == SlotModeEncr)
    se[slotId][sessionId].limitCounter =
       limitValue(
          ss[slotId].se[sessionId].config.encryptConfig.rkEncrMode.limit)
else if (ss[slotId].slotMode == SlotModeDecr)
    se[slotId][sessionId].limitCounter =
       limitValue(
          ss[slotId].se[sessionId].config.decryptConfig.rkDecrMode.limit);

return ErrOk;
```

Предварительные условия

- Сеанс **сегмента AS** успешно инициализирован.

При работе в режиме TS **безопасный видеотракт** сигнализирует соответствующему **клиенту ECI** о замене текущего/следующего управляющего слова (см. [ITU-T J.1012]). **Клиент ECI** может использовать это сообщение для запуска вычисления следующего управляющего слова.

Хост ECI может остановить сеанс и, как следствие, прервать все ожидающие **соединения экспорта** из этого сеанса, используя следующую функцию:

```
int reqAsStopSession(uint slotId, uint sessionId)
```

Семантика

Выполняется следующий код на языке C:

```
int i, j;

ss[slotId].se[sessionId].active = false;
/* рассоединение любых спаренных сеансов дешифрования */
for (j=0; j<NSESSESIONS; j++)
    if (ss[slotId].se[j].coupledSessionId == sessionId)
        ss[slotId].se[j].coupledSessionId = -1;
        /* информирование безопасного видеотракта о рассоединении */

/* отмена всех сеансов экспорта */
if (ss[slotId].slotMode == SlotModeDecr)
    for (i=0; i<NSLOTS; i++)
        for (j=0; j<NSESSIONS; j++)
            if (ss[i].se[j].importSlot == slotId &&
                ss[i].se[j].importSession == sessionId)
            {
                for (k=0; k<MaxExpGrpId; k++)
                    ss[i].se[j].importPermitted[k]= false;
                ss[i].se[j].importSlotId= -1;
                ss[i].se[j].importSession= -1;
            }

return ErrOk;
```

Предварительные условия

- Сеанс **сегмента AS** успешно инициализирован.

8.2.4.4 Управление экспортом сегмента AS

Механизм аутентификации экспорта позволяет **хосту ECI** создавать **соединение экспорта** между сеансом **сегмента AS** в дешифровывающем **клиенте ECI** и сеансом **сегмента AS** в **микросервере**, позволяя таким образом передавать контент из дешифровывающего **клиента ECI** в **микросервер**. Для обработки необходимых цепочек экспорта, импорта и аутентификации экспорта **система AS** использует **подсистему обработки сертификатов** и сеансы экспорта **сегмента AS** POPK и minClientVersion для проверки цепочки экспорта и последующей цепочки импорта. Итогом является положительный результат проверки элемента "**Соединение экспорта**" идентификатора группы экспорта или отказ в соединении. Фактическое соединение осуществляется из сеанса (экспорта) в сеанс импорта.

```
int reqAsExportConnSetup(uint slotId, uint sessId, uint impSlotId,
        uint impSessId, uint grpIndx, CertSerialChain expCh,
        CertSerialChain impCh, CertSerialChain auth[])
```

Семантика

expCh – это цепочка экспорта из POPK в TPEGC или **сертификат** ESC. ImpCh – это цепочка импорта из TPEGC в ESC.

ПРИМЕЧАНИЕ. – impCh может быть пустым. auth [] – это последовательность цепочек аутентификации экспорта, необходимых для участков совместной аутентификации цепочки импорта.

Определение структуры CertSerialChain приведено в пункте 9.5.2.4.2 [ITU-T J.1012].

Сначала **сегмент AS** проверяет impCh, используя цепочки аутентификации экспорта auth[] и установленные **корневой ключ ECI** и номер версии **списка аннулированных сертификатов**.

Затем **сегмент AS** направляет в **подсистему обработки сертификатов** запрос на обработку цепочки экспорта и импорта с использованием POPK, а AS State регистрирует **POPK** и **ExportRlVersion** в качестве корневых. Идентификатор первого **сертификата** в цепочке экспорта сохраняется в переменной expGrpId.

При успешной аутентификации элемент экспорта добавляется к состоянию сеанса **сегмента AS**, содержащему идентификатор группы экспорта и идентификатор сегмента вместе с идентификатором сеанса аутентифицированного экспортирующего **клиента ECI**. Для обработки в целях создания соединения импорта выполняется следующий код на языке C. Аутентификация может быть выполнена для двух идентификаторов групп экспорта, чтобы обеспечить плавный переход в свойствах контента от одной группы экспорта к следующей.

```
/* при успешной обработке цепочек импорта и экспорта CPS предоставляет следующие переменные */
PubKey impSpk;           /* spk импортирующей системы */
uint   impConfigVersion; /* конфигурация номера версии системы экспорта */
uint   expGrpId;         /* группа экспорта, для которой действительно соединение экспорта */


/* проверка на предмет нахождения потенциального сегмента импорта в подходящем состоянии */
if (!( ss[impSlotId].slotMode == SlotModeEncr &&
       ss[impSlotId].se[impSessId].active &&
       ss[impSlotid].se[impSessId].spk == impSpk
       ss[impSlotId].se[impSessId].encryptConfig.microServerVersion >=
         impConfigVersion
   ) ) return ErrExportSlotBadState;
}

/* проверка на предмет существования другого соединения импорта */
if (ss[impSlotId].se[impSessId].ies.importSlotId != ImportNone)
  return ErrExportOngoing;
/* Устанавливает состояние импорта/экспорта сеанса импорта с учетом соединения экспорта */
ss[impSlotId].se[impSessId].ies.importSlotId = slotId;
ss[impSlotId].se[impSessId].ies.importSession = sessId;
ss[impSlotId].se[impSessId].ies.expGrpId[grpIndx] = expGrpId;
ss[impSlotId].se[impSessId].ies.importPermitted[grpIndx] = true;
return ErrOk;
```

Предварительные условия

- Сеанс **сегмента AS** успешно инициализирован.

После установки **соединения экспорта** оно также может быть прервано импортирующей стороной (что фактически останавливает сеанс шифрования):

```
int reqAsExportConnEnd(uint slotId, uint sessionId)
```

Семантика

Выполняется следующий код на языке C:

```
if (!(ss[slotId] != SlotModeEncr)) return ErrImportSlotBadState;
if (!(ss[slotId].se[sessionId].active)) return ErrParam2;
if (ss[slotId].se[sessionId].ies.slotId == -1) return ErrNoExport;

ss[slotId].se[sessionId].ies.importSlotId = -1;
ss[slotId].se[sessionId].ies.importSession = -1;
for (int i=0; i< MaxExpGrpId; i++)
  ss[slotId].se[sessionId].ies.importPermitted[i] = false;
return ErrOk;
```

Предварительные условия

- Сеанс **сегмента AS** установлен для импорта**.**

8.2.4.5 Инициализация лестницы ключей LK1

Для выполнения операций механизма **лестницы ключей в сегменте AS хост ECI** может загрузить ключ связи верхнего уровня LK1 для последующих вычислений результата **лестницы ключей**.

```
int reqAsLoadLk1(uint slotId, uint sessId, InputV inputV,
             ulong spkUri, uchar spkIndx)
```

Семантика

Выполняется следующий код на языке C:

```
if (ss[slotId].slotMode == SlotModeEncr) spkIndx = 0;
if (spkIndx >= 16) return ErrParam5;
/* проверка принадлежности spkUri к set_1 */
if ((spkUri>>spkIndx & 0b1) != 0b1) return ErrSpkUriViolation;
if (!ss[slotId].se[sessId].active) return ErrParam2;
if (spkIndx==0 && ss[slotId].slotMode==SlotModeDecr &&
    ss[slotId].se[sessId].config.decryptConfig.spk0NoDecrypt)
  return ErrSpk0NoDecrypt;

ss[slotId].se[sessId].spkUri = spkUri;
ss[slotId].se[sessId].spkIndx = spkIndx;

if (ss[slotId].slotMode == slotModeEncr &&
    ss[slotId].se[sessId].config.encryptConfig.asymKlMode
){
   ss[slotId].lk1= rnd128();
   return ErrOk;
}

ss[slotId].se[sessId].lk1 =
    blockV_blockC_keyladder(inputV,ss[slotId].se[sessId].spk);
return ErrOk;
```

Предварительные условия

- Сеанс **сегмента AS** инициализирован.

8.2.4.6 Вычисления управляющего слова шифрования

Когда поле состояния **сегмента AS** lk1 установлено, можно вычислить управляющие слова. Значение cwIndx указывает на то, каким является вычисляемое управляющее слово – четным или нечетным. Оно может быть равно 0 (четное) или 1 (нечетное), и для дешифрования на основе файлов оно всегда равно 0.

```
int reqAsComputeEncrCw(uint slotId, uint sessId, ulong cwUri, uint nElk,
    SymKey elk[24], uchar XT[32], uint rkIndx, Field2 field2,
    uint cwIndx)
```

Семантика

Выполняется следующий код на языке C:

```
PubKey spk[MaxSpkEncr+1], popk[MaxSpkEncr+1]; /* временные переменные */
SessionConfig config[MaxSpkEncr+1]; /* временная переменная */

/* основные проверки соответствия */
if (!ss[slotId].se[sessId].active) return ErrParam2;
if (ss[slotId].slotMode != SlotModeEncr) return ErrSlotMode;
if (ss[slotId].se[sessId].config.encryptConfig.rkEncrMode.mode==0b00) {
    if (nElk<2) return ErrParam4;
} else {
    If (nElk<3) return ErrParam4;
}

/* проверка на предмет аутентификации конфигурации сегмента */
if (ss[slotId].se[sessId].configAuthMode != ConfigAuthModeAk1)
    return ErrNoConfigAuth;

/* проверка на предмет нахождения CPS ECI Host Root в достаточном для продолжения состоянии */
if (!cpsEciRootStateOk(slotId,sessId)) return ErrRevocEnforce;


/* проверка на предмет необходимости применять случайный ключ сегмента-сеанса */
SymKey rkAppl; /* случайный ключ, применение которого может потребоваться */
if (rkIndx == 0) {
        rkAppl = ss[slotId].se[sessId].rkState.rkCurrent;
} else if (rkIndx == 1) {
        rkAppl = ss[slotId].se[sessId].rkState.rkNext;
} else {
        return ErrParam7;
}

/* если требуется, вставить случайный ключ сегмента и случайный ключ сеанса  */
if (ss[slotId].se[sessId].config.encryptConfig.rkKlMode) {
        elk[0] = ss[slotId].slotRk;
}
if (ss[slotId].se[sessId].config.encryptConfig.rkEncrMode.mode != RKModeNone) {
        if (nSpk < 3) return ErrNoSlotRkInsert;
        elk[nSpk-1] = rkAppl;
}

/* вычисление input-C и его ввод в лестницу ключей */
uchar result1[16], seField1[16];
ushort cpMask;

computeField1Encrypt(elk[nElk-2], result1, cpMask,
ss[slotId].se[sessId].config.encryptConfig);
computeInputC(result1, field2, elk[nElk-2]);

/* применение ARK со значением 0 */
uchar ark[16] = (uchar){ ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/* определение входных данных spk, popk и config для лестницы ключей; использование spk/popk
сегмента в позиции 0 и репликация конфигурации сегмента */

spk[0]  = ss[slotId].se[sessId].spk;
popk[0] = ss[slotId].popk;
config[0] = ss[slotId]. se[sessId].config;
int i;
int nSpk = slot[slotId]. se[sessId].config.EncryptConfig.nEncr + 1;
for (i=0; i<nSpk-1; i++) {
  spk[i+1]  = ss[SlotId]. se[sessId].encrSpk[i];
  popk[i+1] = ss[SlotId]. se[sessId].decrSpk[i];
  config[i+1] = ss[slotId]. se[sessId].config;
}

/* определение значения spkUri */
```

```
ulong spkUri = (0x1<<(nSpk+1)) - 1; /* для ключей декодирования могут использоваться все SPK */

/* выполнение вычисления лестницы ключей */
bool asym = ss[slotId].se[sessId].config.encryptConfig.asymKlMode;
Secret SymKey cw =
  KeyLadder(ss[slotId].se[sessId].lk1, ss[slotId].se[sessId].encrCwUri,
            AcfCw1Mode, ark, popk, config XT, ss[slotId].spkUri, nSpk, spk,
            nElk,  elk, asym);

/* cw вместе с cwUri, msField1, cpMask и cwIndx направляются в ресурс шифрования */
return ErrOk;
```

Предварительные условия

- Ключ сеанса LK1 загружен.
- При необходимости сеанс **сегмента AS** аутентифицируется.

8.2.4.7 Вычисление управляющего слова дешифрования

Когда поле состояния **сегмента AS** lk1 установлено, можно вычислить управляющие слова. Значение cwIndx указывает на то, каким является вычисляемое управляющее слово – четным или нечетным. Оно может быть равно 0 (четное) или 1 (нечетное), и для дешифрования на основе файлов оно всегда равно 0.

```
int reqAsComputeDecrCw(uint slotId, sessionId, ulong cwUri, uint nSpk,
    uint nElk, SymKey elk[24], PubKey spk[16], PubKey popk[16], SSConflig config[16], uchar
    XT[32], uint rkIndx, Field2 field2, uint cwIndx)
```

Семантика

Выполняется следующий код на языке C:

```
/* основные проверки соответствия */
if (!ss[slotId].se[sessionId].active) return ErrParam2;
if (ss[slotId].slotMode != SlotModeDecr) return ErrSlotMode;
if (ss[slotId].se[sessionId].spkIndx >= nSpk) return ErrParam4;
if (ss[slotId].se[sessionId].config.decryptConfig.rkDecrMode.mode==0b00) {
    if (nElk<2) return ErrParam5;
} else {
    if (nElk<3) return ErrParam5;
}
uint si = ss[slotId].se[sessionId].spkIndx ;

/* проверка на предмет аутентификации конфигурации сегмента, если это требуется */
if ( ss[slotId].se[sessionId].config.decryptConfig.akModeAuth &&
     ss[slotId].se[sessionId].configAuthMode != ConfigAuthModeAk1
   ) return ErrNoConfigAuth;

/* проверка того, находится ли CPS ECI Host Root в достаточном для продолжения состоянии */
if (!cpsEciRootStateOk(slotId,sessionId)) return ErrRevocEnforce ;

/* обеспечение применения подходящих параметров сегмента spk, popk и slotConfig  */
spk[si]= ss[slotId].se[sessionId].spk;
popk[si] = ss[slotId].se[sessionId].popk;

/* аутентификация конфигурации дешифрования сегмента – только при необходимости */
if ( ss[slotId].se[sessionId].config.decryptConfig.klModeAuth )
    ssConfig[si].decryptConfig = ss[slotId].ssConfig.decryptConfig;

/* аутентификация полей klModeAuth и akModeAuth во всех случаях */
config[si].decryptConfig.klModeAuth=
     ss[slotId].se[sessionId].config.decryptConfig.klModeAuth;
config[si].decryptConfig.akModeAuth =
     ss[slotId].se[sessionId].config.decryptConfig.akModeAuth;

/* проверка необходимости применения случайного ключа сегмента-сеанса */
SymKey rpAppl; /* случайный ключ, применение которого может потребоваться */
if (rkIndx == 0) {
        rpAppl = ss[slotId].se[sessionId].rkState.rkCurrent;
} else if (rkIndx == 1) {
        rpAppl = ss[slotId].se[sessionId].rkState.rkNext;
} else {
        return ErrParam11;
}

/* вставить случайный ключ сегмента и случайный ключ сеанса, если требуется  */
if (ss[slotId].se[sessionId].config.decryptConfig.rkKlMode) {
        elk[0] = ss[slotId].slotRk;
```

```
}
if (ss[slotId].se[sessionId].config.decryptConfig.rkDecrMode.mode != RKModeNone) {
        if (nSpk < 2) return ErrNoSlotRkInsert;
        elk[nSpk-2] = rpAppl;
}

/* вычисление input-C, то есть elk[nElk-1] для аутентификации свойств контента */
/* проверка установления бита управления basicUri */
if (((elk[nElk-1][0]>>2)&0b1) != 0b1) return ErrBasicUriCtrl;
uchar result1[16];
computeField1Decrypt(elk[nElk-2],result1);
computeInputC(result1, field2, elk[nElk-2]);

/* применение ARK со значением 0 */
uchar ark[16] = (uchar){0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/* выполнение вычисления лестницы ключей */
Secret SymKey cw =
  KeyLadder(ss[slotId].se[sessionId].lk1, cwUri, AcfCw1Mode, ark,
            popk, ssConfig, XT, ss[slotId].se[sessionId].spkUri, nSpk,
            spk, nElk, elk, false);

/* cw передается в сеанс ресурса дешифрования вместе с cwUri, result1, cwIndx и значением
дескриптора медиаконтента состояний сеанса */

return ErrOk;
```

Предварительные условия

- Ключ сеанса LK1 загружен.
- При необходимости сеанс **сегмента AS** аутентифицируется.

8.2.4.8 Вычисление и применение параметра akClient

Механизм аутентификации блока лестницы ключей позволяет безопасно вычислять с помощью **механизма аутентификации** защищенные ключи для использования **клиентом ECI**:

```
int reqAsComputeAkClient(uint slotId, InputV inputV, uint nSpk,
    uchar spkIndx, PubKey spk[16], PubKey popk[16], SessionConfig akCnf[16],
    ulong spkUri,uchar XT[32], bool online)
```

Семантика

Выполняется следующий код на языке C:

```
/* основные проверки соответствия */
if (ss[slotId].slotMode==SlotModeEncr) spkIndx = 0;
if (spkIndx >= 16) return ErrParam4;
/* проверка принадлежности spkUri к set_1 */
if ((spkUri>>spkIndx & 0b1) != 0b1) return ErrSpkUriViolation;
if (ss[slotId].slotMode == SlotModeEncr) {
  if (akCnf[spkIndx].encryptConfig.configVersion != 0x1) return ErrParam7;
  if (akCnf.encryptConfig.microServerVersion >
      ss[slotId].clientPOClRLVnr) return ErrRevocEnforce;
  if ((cpsEciRootState.rootVersion <
      akCnf[spkIndx].encryptConfig.minEciRootState.rootVersion)
   || (cpsEciRootState.rlVersion <
      akCnf[spkIndx].encryptConfig.minEciRootState.rlVersion))
    return ErrRevocEnforce;
}
if (ss[slotId].slotMode == SlotModeDecr) {
  if (akCnf[spkIndx].decryptConfig.configVersion != 0x1) return ErrParam7;
  if (akCnf.decryptConfig.minClientVersion >
      ss[slotId].clientPOClRLVnr) return ErrRevocEnforce;
  if ((cpsEciRootState.rootVersion <
      akCnf[spkIndx].decryptConfig.minEciRootState.rootVersion)
   || (cpsEciRootState.rlVersion <
      akCnf[spkIndx].decryptConfig.minEciRootState.rlVersion))
    return ErrRevocEnforce;
}
/* обеспечение применения подходящих параметров сегмента spk и popk */
popk[spkIndx] = ss[slotId].popk;

/* обеспечение применения подходящих ACF и ARK */
uchar ark[16] ;
uchar acf[15] = acfAk1Mode ;
acf[1] = AkUseCl;
```

```
if (online) {
acf[1] += AkOnline;
ark = ss[slotId].slotRk;
} else {
    acf[1] += AlOffline;
    ark = {0} ;
}

/* выполнение механизма аутентификации */
ss[slotId].akClient =
      AuthMech(inputV,acf,ark,popk,akCnf,XT,spkUri,nSpk,spkIndx,spk);
return ErrOk;
```

Предварительные условия

- Сегмент инициализирован.

Для использования вычисленного ключа AK **клиента ECI** определена следующая функция:

```
int reqAsClientChalResp(int slotId, uchar challenge[16],
    uchar *(response[16]));
```

Семантика

Выполняется следующий код на языке C:

```
*response = AuthMechResponse(ss[slotId].akClient, challenge);
return ErrOk;
```

Предварительные условия

- Сегмент инициализирован.
- Успешно вычислено значение AkClient сегмента.

8.2.4.9 Аутентификация конфигурации сеанса сегмента AS

Механизм аутентификации блока лестницы ключей позволяет **серверу подготовки** аутентифицировать конфигурацию сеанса сегмента. **Сервер подготовки** может выдать информацию аутентификации в автономном режиме или потребовать онлайн-аутентификации, установив флаг AkOnline в ACF. Для аутентификации сегментов шифрования и дешифрования имеются две отдельные функции.

```
int reqAsAuthDecrConfig(uint slotId, uint sessId, InputV inputV,
  uint nSpk, uchar spkIndx, PubKey spk[16], PubKey popk[16], SSCnfg clCnf[16],
  ulong spkUri, uchar XT[32], bool online, uchar verifier[16])
```

Семантика

Выполняется следующий код на языке C:

```
/* основные проверки соответствия */
if (!ss[slotId].se[sessionId].active) return ErrParam2;
if (ss[slotId].slotMode!=SlotModeDecr) return ErrSlotMode;
if (spkIndx >= 16) return ErrParam5;
/* проверка принадлежности spkUri к set_1 */
if ((spkUri>>spkIndx & 0b1) != 0b1) return ErrSpkUriViolation;
if (spkIndx==0 && ss[slotId].slotMode==SlotModeDecr &&
    ss[slotId].se[sessId].config.decryptConfig.spk0NoDecrypt) return ErrSpk0NoDecrypt;

/* проверка того, находится ли CPS ECI Host Root в достаточном для продолжения состоянии */
if (!cpsEciRootStateOk(slotId)) return ErrRevocEnforce;

/* обеспечение применения подходящих параметров сегмента spk, popk и config  */
popk[spkIndx] = ss[slotId].popk;
spk[spkIndx] = ss[slotId].se[sessId].spk;
clCnf[spkIndx] = ss[slotId].se[sessId].config;

uchar ark[16];
uchar acf[15] = acfAk1Mode;
acf[1] = AkUseAS + AkConfigAuth;
if (online) {
        acf[1] = AkOnline;
        ark = ss[slotId].slotRk;
} else {
        acf[1] = AkOffline;
        ark = {0};
}
```

```
/* выполнение механизма аутентификации */
Secret SymKey ak =
      AuthMech(inputV,acf,ark,popk,clCnf,XT,spkUri,nSpk,spkIndx,spk);

uchar response[16] = AuthMechResponse(ak, verifier);

if (response == {0}) {
        ss[slotId].se[sessId].configAuthMode = ConfigAuthModeAk1;
        return ErrOk;
} else {
        ss[slotId].se[sessId].configAuthMode = ConfigAuthModeNone;
        return ErrSlotConfigAuthFail;
}
```

Предварительные условия

- Ключ сеанса LK1 **сегмента AS** загружен**.**

Аутентификация при шифровании включает проверку конкретного состояния шифрования.

```
int reqAsAuthEncrConfig(uint slotId, uint sessId, InputV inputV,
       uchar XT[32], bool online, uchar verifier[16])
```

Семантика

Выполняется следующий код на языке C:

```
PubKey spk[MaxSpkEncr+1], popk[MaxSpkEncr+1]; /* временны́е переменные */
SessionConfig config[MaxSpkEncr+1]; /* временна́я переменная */

/* основные проверки соответствия */
if ((ss[slotId.SlotMode != SlotModeEncr) return ErrSlotMode;

/* проверка того, находится ли CPS ECI Host Root в достаточном для продолжения состоянии */
if (!cpsEciRootStateOk(slotId,sessId)) return ErrRevocEnforce;

/* определение входных данных spk, popk и config для лестницы ключей; использование spk/popk
сегмента в позиции 0 и репликация конфигурации сегмента */

spk[0]  = ss[slotId].se[sessId].spk;
popk[0] = ss[slotId].popk;
config[0] = ss[slotId].se[sessId].config;
int i;
int nSpk = slot[slotId]. config.EncryptConfig.nEncr + 1;
for (i=0; i<nSpk-1; i++) {
  spk[i+1] = ss[SlotId].encrSpk[i];
  popk[i+1] = ss[SlotId].encrPopk [i];
  config[i+1] = ss[slotId].se[sessId].slotConfig;
}

/* определение значений spkUri */
ulong spkUri = (0x1<<(nSpk)) - 1;
                      /* для декодирования контента могут использоваться все SPK */

uchar ark[16];
uchar acf[15] = acfAk1Mode;
acf[1] = AkUseAS + AkConfigAuth;
if (online) {
        acf[1] = AkOnline;
        ark = ss[slotId].slotRk;
} else {
        acf[1] = AkOffline;
        ark = {0};
}

/* выполнение механизма аутентификации */
Secret SymKey ak =
      AuthMech(inputV,acf,ark,popk,clCnf,XT,spkUri,nSpk,spkIndx,spk);

uchar response[16] = AuthMechResponse(ak, verifier);

if (response == {0}) {
        ss[slotId].se[sessId].configAuthMode = ConfigAuthModeAk1;
        return ErrOk;
} else {
        ss[slotId].se[sessId].configAuthMode = ConfigAuthModeNone;
        return ErrSlotConfigAuthFail;
}
```

Предварительные условия

- Ключ сеанса LK1 **сегмента AS** загружен.

8.2.4.10 Загрузка секретного ключа микросервера

Клиент **микросервера** может использовать сегмент AS, работающий в режиме асимметричного сервера, и загружать значение секретного ключа **микросервера** ussk для последующего установления безопасного соединения с чипсетом **микроклиента**:

```
int reqAsLdUssk(uint slotId, uint sessId, InputV inputV,
                uchar XT[32], bool online, uchar mUssk[NUSSK])
```

Семантика

Выполняется следующий код на языке C:

```
PubKey spk[MaxSpkEncr], popk[MaxSpkEncr];
SessionConfig config[MaxSpkEncr];

/* основные проверки соответствия */
if (ss[slotId].slotMode!=SlotModeEncr) return ErrSlotMode;
if (!ss[slotId].se[sessId].config.encryptConfig.asymKlMode)
        return ErrSlotModeUndefined;

/* проверка того, находится ли CPS ECI Host Root в достаточном для продолжения состоянии */
if (!cpsEciRootStateOk(slotId,sessId)) return ErrRevocEnforce;

spk[0]  = ss[slotId].se[sessId].spk;
popk[0] = ss[slotId].popk;
config[0] = ss[slotId]. se[sessId].config;
int i;
int nSpk = slot[slotId]. se[sessId].config.EncryptConfig.nEncr + 1;
for (i=0; i<nSpk-1; i++) {
  spk[i+1]  = ss[SlotId]. se[sessId].encrSpk[i];
  popk[i+1] = ss[SlotId]. se[sessId].decrSpk[i];
  config[i+1] = ss[slotId]. se[sessId].config;
}
/* определение значений spkUri */
ulong spkUri = (0x1<<(nSpk+1)) - 1; /* для ключей декодирования могут использоваться все SPK */

uchar ark[16];
uchar acf[15] = acfAk1Mode;
acf[1] = AkUseAS + AkLdUssk;
if (online) {
        acf[1] = AkOnline;
        ark = ss[slotId].slotRk;
} else {
        acf[1] = AkOffline;
        ark = {0};
}

/* выполнение механизма аутентификации */
Secret SymKey ak =
      AuthMech(inputV,acf,ark,popk,config,XT,spkUri,nSpk,0,spk);

/* выполнение декодирования ussk с использованием ECB AES */
int i,j;
uchar response[32];
for (i=0; i<NUSSK; i+=32){
response = AuthMechResponse(ak, &(mUssk[i]));
for (j=0; j<32; j++) ss[slotId].se[sessId].ussk[i+j] = response[j];
}
return ErrOk;
```

Предварительные условия

- Конфигурация сеанса аутентифицирована.

8.2.4.11 Генерирование MinitLk1 для микроклиентов

В режиме асимметричного микросервера **AS Slot** может генерировать сообщения инициализации **блока лестницы ключей** для **микроклиентов**:

```
InputV reqAsMInikLk1(uint slotId, uint sessId, ECI_Certificate_Chain ClCPK)
```

Семантика

Функция ECI_Certificate_Chain определена в пункте 5.4.1 [ITU-T J.1012] и содержит **цепочку сертификатов** для проверки **микроклиента**. Сначала она применяет CPS к первому сертификату в цепочке для проверки CLCPK, используя параметр slot[slotId] с ключом POPK в качестве **родительского сертификата** и ss[SlotId].se[sessId].config.encryptConfig.microServerVersion в качестве минимального номера версии **списка аннулированных сертификатов**. Если эта проверка прошла успешно, переменная clcpk содержит открытый ключ чипсета клиента, и выполняется следующий код на языке C:

```
return asymInitLk1(ss[slotId].lk1, slot[slotId].ussk, clcpk);
```

Предварительные условия

- Ключ Ussk инициализирован.
- Сеанс находится в режиме асимметричного шифрования.

8.2.4.12 Вычисление ключа дешифрования образа клиента ECI

Для выполнения загрузки шифрованного образа **сегмент AS** может предоставить ключ аутентификации, с помощью которого можно дешифровать ключ для декодирования образа. До инициализации сегмента выполняется следующая функция:

```
int reqAs ComputeImageKey(uint slotId, InputV inputV,
        symKey eKey , bool online, ECIRootState min_root_state)
```

Семантика

Выполняется следующий код на языке C:

```
/* используется состояние конфигурации сегмента по умолчанию */
SessionConfig config = {
  .decryptConfig = {
    .configVersion = 0x1,
    .reserved1 = 0x0,
    .klModeAuth = 0x0,
    .akModeAuth = 0x0,
    .rkKlMode = 0x0,
    .spk0NoDecrypt = false,
    .reserved2 = 0b000000,
    .rkDecrMode = { 0 },
    .minEciRootState = min_root_state,
    .expRlVersion = 0x0
  },
  .encryptConfig = { 0 }
} ;

if (!(cpsEciRootState.rootVersion >= min_root_state.rootVersion &&
     (cpsEciRootState.rlVersion >= min_root_state.rlVersion))
    return ErrRevocEnforce;

/* непосредственное создание popk/spk, XT, clCnf, */
PubKey popkArr[1]; /* также используется для spk */
popkArr[0] = ss[slotId].popk;
SessionConfig cnf[1];
cnf[0] = config;
uchar XT[32] = {0};
ulong spkUri= 0x1;

uchar ark[16];
uchar acf[15] = acfAk1Mode;
acf[1] = AkUseAS + AkClImg;
if (online) {
        acf[1] = AkOnline;
        ark = ss[slotId].slotRk;
} else {
        acf[1] = AkOffline;
        ark = {0} ;
}

/* выполнение механизма аутентификации */
Secret SymKey ak =
      AuthMech(inputV,acf,ark,popkArr,cnf,XT,spkUri,1,0,popkArr);

Secret SymKey dImgKey = AuthMechResponse(ak, eImgKey);
```

```
/* впоследствии dImgKey используется загрузчиком клиента для дешифрования образа клиента в режиме
CBC AES при IV=0 */

return ErrOk;
```

Предварительные условия

- Сегмент установлен в состояние по умолчанию; slotRk присвоено новое случайное значение.

ПРИМЕЧАНИЕ. – Эта функция по запросу **клиента ECI** не выполняется.

8.2.4.13 Считывание информации усовершенствованной системы безопасности

Система AS предоставляет **клиенту ECI** доступ к генерируемым ею данным и обеспечивает функцию генерирования случайного ключа общего назначения для **клиента ECI**.

ПРИМЕЧАНИЕ 1. – Функции get и set, определенные в этом разделе, не генерируют автоматические сообщения об ошибке при неопределенных значениях параметров; функции get просто возвращают неопределенное значение, а функции set не оказывают никакого действия.

Следующая функция считывает случайный ключ **сегмента AS** (обычно используется в качестве одноразового кода для сеансов):

```
SymKey getAsSlotRk(uint slotId)
```

Семантика

Выполняется следующий код на языке C:

```
return ss[slotId].slotRk;
```

Если сегмент не инициализирован, число возвращается.

Следующая функция считывает состояние случайного ключа сеанса:

```
SymKey getAsSessionRk(uint slotId, uint sessionId, uint rkIndx)
```

Семантика

Выполняется следующий код на языке C:

```
if (rkIndx == 0)
  return se[slotId][sessionId].rkState.rkCurrent;
else
  return se[slotId][sessionId].rkState.rkNext;
```

Если сегмент не инициализирован, число возвращается.

Можно считать значение счетчика ограничения случайного ключа сеанса:

```
ulong getAsSessionLimitCounter (uint slotId, uint sessionId)
```

Семантика

Выполняется следующий код на языке C:

```
return se[slotId][sessionId].rkState.limitCounter;
```

Если сегмент не инициализирован, число возвращается.

Можно установить значение ограничительного счетчика, при котором происходит то или иное событие (например, обновление случайного ключа через достаточный промежуток времени):

```
ulong setAsSessionLimitEvent(uint slotId, uint sessionId, ulong eventLimit)
```

Семантика

Событие eventSessionLimitCounter происходит, когда после вызова этой функции выполняется следующее условие:

```
se[slotId][sessionId].rkState.limitCounter <= eventLimit;
```

ПРИМЕЧАНИЕ 2. – Второй вызов отменяет предыдущий. Вызов этой функции во второй раз с очень большим значением eventLimit фактически отменяет событие (за исключением случая, когда оно уже произошло).

При достижении предела события для сеанса происходит следующее событие:
```
reqAsEventSessionLimit(uint slotId, uint sessionId)
```

ПРИМЕЧАНИЕ 3. – Согласно [ITU-T J.1012] это событие переводится в асинхронное сообщение без соответствующего ответа.

8.2.4.14 Генерирование случайных чисел для клиента

Клиент ECI может запросить 128-битовое случайное число, сгенерированное **системой AS**, вызвав следующую функцию:
```
SymKey getAsClientRnd()
```

Семантика

Выполняется следующий код на языке C:
```
return rnd128();
```

8.2.4.15 Коды ошибок

В таблице 8-14 приведены значения кодов ошибок, которые возвращаются функцией, определенной в пункте 8.2.4.

Эти коды ошибок соответствуют соглашению о кодах ошибок в сообщениях между **хостом ECI** и **клиентом ECI**, приведенному в разделе 9 [ITU-T J.1012].

Таблица 8-14 – Определение кодов возвращения ошибок

Код возвращения ошибки	Значение	Описание
ErrSlotMode	–256	**Сегмент AS** находится в режиме, не подходящем для этой операции
ErrNoMoreSessions	–257	Сеанс больше не существует
ErrSession1Coupled	–258	Первый сеанс уже спарен
ErrSession2Coupled	–259	Второй сеанс уже спарен
ErrSessionNotCoupled	–260	Сеанс не спарен
ErrNoSuchSession	–261	Сеанса не существует
ErrExportNoSlot	–262	Сегмент экспорта неизвестен
ErrExportSlotBadState	–263	Сегмент экспорта находится в ненадлежащем состоянии
ErrExportOngoing	–264	В сеансе экспорта уже есть **соединение экспорта**
ErrImportSlotBadState	–265	Сегмент импорта не находится в режим шифрования
ErrNoExport	–266	Экспорт в сеансе не происходит
ErrSpkUriViolation	–267	Неправильное значение SpkUri SPK сегмента для режима сегмента
ErrSlotModeUndefined	–268	Неправильное значение режима сегмента для данной операции
ErrRevocEnforce	–269	**Принцип аннулирования сертификатов ECI** не позволяет использовать сегмент
ErrNoConfigAuth	–270	Неправильно аутентифицирована конфигурация сегмента
ErrNoSlotRkInsert	–271	Недостаточно длинный вектор ELK для вставки случайного ключа
ErrSpk0NoDecrypt	–272	spk[0] не может использоваться для генерации управляющих слов дешифрования
ErrBasicUriCtrl	–273	Не установлен управляющий бит field1 базового URI
ErrOk	0	Успешный вызов
ErrSlotConfigAuthFail	–274	Неуспешная аутентификация конфигурации сеанса сегмента
ErrParam<N>	**–<N>**	Ошибка входного параметра **N** (ErrParam1 имеет значение –1 и сигнализирует об ошибке в параметре 1)
	1..MaxInt	Успешный вызов, значение определено в определениях сообщений

9 Скремблирование/дескремблирование и экспорт контента

9.1 Основные функции

Безопасный видеотракт поддерживает дешифрование контента. Этот контент сопровождается **свойствами контента** и **соединениями экспорта**. Контент может быть перенаправлен на стандартные выходы, если это позволяют **свойства контента**, и повторно зашифрован **микросервером** при наличии соответствующего **соединения экспорта**.

Для управления ресурсами **ECI** выделяет ресурсы дешифрования и шифрования. Ресурс используется для дешифрования или шифрования контента одного медиасеанса, зашифрованного или подлежащего шифрованию, с одним CW в каждый отдельно взятый момент времени. Ресурс дешифрования или

шифрования подключается к одному сегменту **AS** дешифрования или шифрования. Для дешифровки потока TS у ресурса дешифрования имеется двойной буфер для нечетного и четного значения CW. Нечетное или четное значение CW выбирается потоком, подлежащим декодированию. Это может привести к необходимости динамического изменения управляющего слова в случае изменения **свойств контента**, подлежащего шифрованию. Для дешифрования и шифрования на основе файлов **хост ECI** обеспечивает синхронизацию CW с контентом, подлежащим дешифровке, которая может происходить значительно быстрее, чем в режиме реального времени. Ресурсам дешифрования и повторного шифрования на основе файлов требуется только один буфер CW.

ПРИМЕЧАНИЕ. – Для потоков TS, требующих двух или более управляющих слов для дескремблирования разных элементарных потоков, нужны несколько ресурсов дескремблирования и, следовательно, несколько сеансов.

В **ECI** не определены никакие детали, связанные с буферизацией или (возможно, обширной) промежуточной обработкой, такой как перекодирование или введение меток, которой может подвергаться дешифрованный контент, передаваемый из ресурса дешифрования в ресурс шифрования. Такая обработка может вызывать значительные задержки. Производители CPE могут выбирать соответствующие реализации, приводящие к смещению по времени между ресурсом дешифрования и связанным с ним ресурсом повторного шифрования. Сегмент повторного шифрования и **клиент ECI** синхронизируются с шифрованием контента.

9.2 Спецификации скремблера и дескремблера

В режиме TS функция дескремблирования CPE **ECI** должна поддерживать следующие алгоритмы дескремблирования:

- CSA1/2, PES и режим TS, определенные в [ETSI ETR 289] и [b-ETSI DVB CSA].
- CSA3, PES и режим TS, определенные в [ETSI TS 100 289] и [b-ETSI DVB CSA3].
- DVB-CISSA PES и режим TS, определенные в [ETSI TS 103 127].

В режиме File функция дескремблирования CPE **ECI** должна поддерживать следующие алгоритмы дескремблирования:

- Режим CENC AES128 CTR и режим AES128-CBC (как полное, так и выборочное шифрование), определенные в [ISO/IEC 23001-7]. CENC и [ISO/IEC 23009-4] для MPEG-DASH.

В режиме TS функция скремблирования CPE **ECI** должна поддерживать следующие алгоритмы скремблирования:

- DVB-CISSA PES и режим TS [ETSI TS 103 127].

В режиме File функция скремблирования CPE **ECI** должна поддерживать следующие алгоритмы скремблирования:

- Режимы CENC AES128 CTR и CBC (как полное, так и выборочное шифрование), определенные в [ISO/IEC 23001-7] и [ISO/IEC 23009-4]. **Безопасный видеотракт** генерирует уникальный вектор инициализации для контента, зашифрованного одним CW для режима AES-CTR, и следовать правилам определения IV для режима AES-CBC, как определено в [ISO/IEC 23009-4]. Для упаковки контента **хост ECI** может обращаться к векторам инициализации.

9.3 Управление экспортом

Аутентифицированные **соединения экспорта** используются сеансами дешифрования **сегмента AS** в качестве билетов для авторизации импорта и экспорта ресурсом дешифрования. Ресурс дешифрования разрешает экспорт дешифрованного контента в ресурс шифрования, если **соединение экспорта**, предоставленное соответствующим сеансом **сегмента AS**, разрешает это для идентификатора группы экспорта и **свойства контента** дешифрованного контента передают соответствующий идентификатор группы экспорта, как определено в пункте 8.2.4.4. Ресурс дешифрования не разрешает экспорт дешифрованного контента в ресурс шифрования, если **соединение экспорта** группы экспорта, выбранное по идентификатору группы экспорта в **свойствах контента**, не является подтвержденным **соединением экспорта**, предоставленным соответствующим **сегментом AS**.

9.4 Управление выходом

Управляющие выходными данными **свойства контента** используют для запрета или разрешения экспорта контента в соответствии со стандартными технологиями защиты выходных данных CPE. Ресурс дешифрования разрешает экспорт дешифрованного контента в выходной файл, если информация управления выходными данными соответствующего сеанса **сегмента AS** позволяет это. Ресурс дешифрования запрещает экспорт дешифрованного контента в выходной файл, если информация управления выходными данными соответствующего сеанса **сегмента AS** этого не позволяет.

9.5 Сравнение свойств контента спаренных сеансов

Безопасный видеотракт проверяет, что **свойства контента**, определенные в field1 сеанса, исключая первые два байта, равны **свойствам контента** любого спаренного сеанса. Разрешается экспорт и вывод контента спаренного сеанса, имеющего равные **свойства контента**. С этого момента с точки зрения защиты **ECI** объединенные потоки рассматриваются как один сеанс. Если **свойства контента** field1, исключая первые два байта, не равны, объединение спаренного потока блокируется.

9.6 Передача свойств контента при экспорте

Сеанс ресурса дешифрования передает **свойства контента** field1, установленные клиентом и неявно (частично) заверенные **лестницей ключей**, вместе с контентом ресурсам сеанса повторного шифрования, импортирующим дешифрованный контент, как определено в пункте 8.2.4.6. Сеансы шифрования, принимающие дешифрованный контент, проверяют указанные байты field1 на предмет соответствия значению, присвоенному field1 для шифрования контента, с применением маски для выбора полей, которые требуется передать, как определено этой функцией, гарантируя таким образом, что указанные дешифруемые байты field1 клиента были переданы в зашифрованный контент.

При каждом изменении входных значений impField1, expField1 и cpMask сеанс ресурса шифрования выполнят следующий код на языке C:

```
uchar impField1[16];  /* значения field1 импортируемого контента */
uchar expField1[16];  /* вычисление значений field1 по CW шифрования */
ushort cpMask;        /* маска сравнения */

bool propOk = true;  /* указывает, что импортированный контент передается правильно */
int i;

for (i=2; i<16; i++)
    propOk &&= !(cpMask>>I & 0b1) || (impField1[i] == expField1[i]);

if (propOk) /* выполняется повторное шифрование контента */
else /* повторное шифрование контента не выполняется  */
```

9.7 Установление базового URI для экспорта

Управление передачей базового URI из сеанса дешифрования **сегмента AS** в сеанс повторного шифрования **сегмента AS** осуществляется с помощью следующих механизмов, в которых slotId идентифицирует сегмент шифрования, а sessionId – сеанс шифрования:

Права, назначенные контенту с базовым URI **сегментом AS** шифрования, не более либеральны, чем права передаваемого контента.

Микросервер аутентифицирован следующим образом: значение ss[slotId].se[sessionId].config. decryptConfig,akModeAuth равно 0b1.

Если базовый URI не разрешает повтор воспроизведения контента (то есть режим потоковой передачи), то при экспорте проверяется следующее:

– значение ss[slotId].se[sessionId].config.decryptConfig.rkDecrMode.mode должно быть не равно значению RKModeNone (то есть применяется случайный одноразовый код, предотвращающий повторное воспроизведение ранее закодированного контента при перезапуске системы); и

– устанавливается значение ss[slotId].se[sessionId].klModeAuth, равное 0b1, что гарантирует, что конфигурация decryptConfig, используемая сервером, включая вставку случайного ключа в **микроклиенте**, аутентифицирована и используется **микроклиентом** на основе вычисления **лестницы ключей**.

9.8 Применение свойств контента на стандартных выходах

На стандартном выходе, который обычно представляет собой физический выход в сочетании со стандартной системой защиты, **свойства контента** используются для выбора подходящих параметров защиты выхода или запрета выхода, если установка подходящих параметров невозможна. Точные правила для этого определены в правилах соответствия.

Уровень **устойчивости** реализации базового URI и **свойств контента** для управления выходом должен быть таким же, что и для **безопасного видеотракта**.

Устойчивость соблюдения стандартного URI должна быть как минимум такой же, как и для реализации **хоста ECI**, за исключением функций со сложными требованиями к реализации.

9.9 Синхронизация управляющего слова

Для обработки потоков TS **безопасный видеотракт** предоставляет следующие функции, позволяющие управлять изменением управляющих слов (для шифрования), и выдает уведомления об изменениях поля управления скремблированием. Функции и события этого сеанса соответствуют соглашениям, определенным в пункте 8.2.4.

Сеанс **сегмента AS** может обеспечить применение как "нечетного", так и "четного" управляющего слова для шифрования или дешифрования контента.

В случае дешифрования поле управления скремблированием [ETSI TS 100 289] сообщает функции дешифрования, какое управляющее слово использовать. Если контент обозначается как нескремблированный, управляющее слово не используется. Значение результата равно значениям полей управления скремблированием, указанным в пункте 5.1 [ETSI TS 100 289].

Текущий статус поля управления скремблированием в потоке считывает следующая функция:
```
uint getAsSC(uint slotId, uint sessionId)
```

В случае шифрования применяемое управляющее слово может изменяться при двух событиях:

1) Изменение **свойств контента** импортируемого контента, которое инициирует изменение управляющего слова, применяемого для шифрования. Это изменение может быть отложено **сегментом AS** для завершения текущего изменения управляющего слова, вызванного следующим событием.

   Поступающий из сеанса **сегмента AS** сигнал о необходимости изменения управляющего слова.

Если импортированный контент не скремблирован, то для шифрования скремблирование не требуется, и в первом возможном месте изменения поле управления скремблированием контента устанавливается в 0b00. И наоборот, если импортируемый контент переводится из нескремблированного в скремблированный, он скремблируется со следующим управляющим словом; выбирается ключ, противоположный по отношению к контенту, скремблированному до начала открытой части контента.

Событие, сигнализирующее об изменении свойств импортируемого контента, определяется следующим образом:
```
reqAsEventCpChange(uint slotId, uint sessionId)
```

Семантика

Событие сигнализирует об изменении **свойств контента** импортируемого контента, если этот контент требует шифрования.

Безопасный видеотракт не должен допускать расхождения между параметрами шифрования и **свойствами контента** импортируемого контента более длительный период времени, чем указано в пункте 6.6.2 [b-ITU-T J Suppl. 7].

ПРИМЕЧАНИЕ 1. – При переходе от шифрованного импортируемого контента к нешифрованному это событие не создается. К нешифрованному контенту **свойства контента** не применяются.

Безопасный видеотракт позволяет **системе AS** задерживать любое автоматическое изменение события **свойств контента** eventCpChange по следующей команде:
```
setAsPermitCPChange(uint slotId, uint sessionId, bool permit)
```

Семантика

Эта функция разрешает автоматическое изменение управляющих свойств импортируемого контента для изменения управляющего слова шифруемого контента.

ПРИМЕЧАНИЕ 2. – Эта функция должна предшествовать любому следующему управляющему слову, вычисленному не на основе последующих **свойств контента**, например отражающему только одноразовый код или изменение случайного ключа.

ПРИМЕЧАНИЕ 3. – Если изменение запрещено (`permit==false`), его разрешение должно быть восстановлено в течение допустимого времени несоответствия со **свойствами контента** импортируемого контента, чтобы не вызвать "отключения" в перешифрованном потоке.

Следующая функция позволяет установить определенный статус поля управления скремблированием при шифровании:

```
setAsSC(uint slotId, uint sessionId, uint scramblingControlField)
```

Семантика

Значение поля управления скремблированием scramblingControlField устанавливается в первой возможной точке изменения потока. Для scramblingControlField допускаются только значения 0b10 и 0b11 (соответственно скремблирование четным и нечетным ключами).

Если импортируемый контент имеет статус незашифрованного, то для поля управления скремблированием зашифрованного потока устанавливается значение 0b00 (без скремблирования).

Для сеансов дешифрования и шифрования определена следующая функция создания события:

```
reqAsEventSC(uint slotId, uint sessionId, uint scramblingControlField)
```

Семантика

Событие создается при изменении статуса поля управления скремблированием.

10 Подсистема обработки сертификатов

10.1 Основные правила обработки цепочек сертификатов

Подсистема обработки сертификатов может обрабатывать **цепочки сертификатов** для аутентификации объектов на основе первоначального открытого ключа и минимального номера версии **списка аннулированных сертификатов**. Бóльшая часть обработки цепочки сертификатов носит общий характер. В этом разделе определяются общие правила обработки **цепочек сертификатов**. В следующих разделах определяются специальные правила обработки цепочек разного типа.

Нижеследующие **цепочки сертификатов** определены в пункте 5.4 [ITU-T J.1012].

При определении правил CPS применяется поэтапный подход к обработке **цепочек сертификатов**, начиная с начала цепочки (первый **список аннулированных сертификатов**) и использования первоначального открытого ключа и минимального номера версии **списка аннулированных сертификатов**. Первым этапом является проверка **списка аннулированных сертификатов**. На втором этапе проверяется следующий **сертификат** в цепочке. После однократного выполнения этапов 1 и 2 определяются новый открытый ключ и новый номер версии **списка аннулированных сертификатов** для обработки оставшейся части цепочки. Этапы 1 и 2 повторяются до тех пор, пока не будет обработана вся цепочка. В общем случае рекомендуется, чтобы программные функции, выдающие цепочки, предварительно проверяли эти цепочки во избежание неожиданного сбоя CPS при обработке цепочки.

Общие этапы обработки **цепочки сертификатов**:

1) CPS выполняет следующую проверку **списка аннулированных сертификатов**:
   a) CPS проверяет поле **format_version списка аннулированных сертификатов** на соответствие версии, которую она может интерпретировать (см. специальные правила обработки цепочек), и поля **rl_id.type** и **rl_id.rl_indicator** на соответствие ожидаемым значениям;

b) если **родительским объектом** является **корневой сертификат** (**root_version_indicator**=1), то **хост ECI** должен выбрать в качестве **родительского объекта корневой сертификат** с версией **root_version**, в противном случае используется предварительно загруженный или предшествующий **сертификат**;

c) CPS проверяет подпись **списка аннулированных сертификатов** с помощью последнего заверенного открытого ключа;

d) CPS проверяет соответствие длины **списка аннулированных сертификатов** значению соответствующего поля и наличие у любого поля переменной длины соответствующей длины;

e) CPS осуществляет проверку того, что номер версии **списка аннулированных сертификатов** не был отменен минимальным номером версии такого **списка**.

2) CPS выполняет следующую проверку **сертификата**:

a) CPS осуществляет проверку того, что ***следующие*** поля <type, entity_ id, version> **сертификата** в цепочке не аннулированы в соответствии с последним **списком аннулированных сертификатов** и устанавливают минимальную версию **списка аннулированных сертификатов**, сопровождающую этот **сертификат**, в соответствии с полями **base_rl_version** и **min_rl_version** последнего **списка аннулированных сертификатов**;

b) CPS проверяет поле **format_version списка аннулированных сертификатов** на соответствие версии, которую разрешено интерпретировать;

c) CPS проверяет соответствие длины **сертификата** значению соответствующего поля и наличие у любого поля переменной длины соответствующей длины;

d) CPS проверяет подпись **сертификата** открытым ключом.

После этапов 1 и 2 обработки открытый ключ и минимальный номер версии **списка аннулированных сертификатов** обновляются. Открытый ключ принимает значение поля открытого ключа **сертификата**, обработанного на этапе 2, а минимальной номер версии **списка аннулированных сертификатов** – значение, полученное на этапе 2a.

Не все **сертификаты** должны сопровождаться списком аннулированных сертификатов. Если старший бит поля типа идентификатора сертификата равен нулю, то для дальнейшей обработки цепочки в **подсистеме обработки сертификатов** требуется, чтобы сертификат сопровождался **списком аннулированных сертификатов**. Если **список аннулированных сертификатов** не требуется, то никакая обработка **списка аннулированных сертификатов** и номера версии, а также номеров версий **списка аннулированных сертификатов** на вышеизложенных этапах не осуществляется.

10.2 Специальные правила для цепочек сертификатов образов хоста

CPS выполняет специальную проверку цепочек сертификатов образов хоста:

1) Первый **список аннулированных сертификатов** относится к типу 0x1 (список аннулированных сертификатов производителя).

2) Первый **сертификат** относится к типу 0x1 (**сертификат** производителя).

3) Второй **список аннулированных сертификатов** относится к типу 0x0 (**список аннулированных сертификатов хоста ECI**).

4) Второй **сертификат** относится к типу 0x0 (**сертификат хоста ECI**).

5) Возможный третий **сертификат** относится к типу 0x98 (**сертификат** серии образов **хоста**).

Для проверки фактического образа **хоста ECI** используется открытый ключ последнего **сертификата** (**сертификата хоста ECI** или **сертификата** серии образов **хоста ECI**).

10.3 Специальные правила для цепочек сертификатов образов клиента

CPS выполняет специальную проверку цепочек сертификатов образов клиента:

1) Первый **список аннулированных сертификатов** относится к типу 0x2 (**список аннулированных сертификатов** поставщика).

2) Первый **сертификат** относится к типу 0x2 (**сертификат** поставщика).

3) Второй **список аннулированных сертификатов** относится к типу 0x0 (**список аннулированных сертификатов клиентов ECI**).

4) Возможный второй **сертификат** относится к типу 0x1 (**сертификат** серии клиентов).

Открытый ключ последнего **сертификата** (**сертификат** поставщика либо сертификат серии клиентов) используется для проверки фактического образа **хоста ECI** с учетом последнего номера версии клиентского **списка аннулированных сертификатов** для проверки версии образа в том случае, если последний **сертификат** – это **сертификат** поставщика.

10.4 Специальные правила для сертификатов системы управления платформой

CPS применяет специальную проверку **цепочек сертификатов системы управления платформой**:

1) Первый **список аннулированных сертификатов** относится к типу 0x3 (список аннулированных сертификатов **оператора**).

2) Первый **сертификат** относится к типу 0x3 (**сертификат оператор**а).

3) Второй **список аннулированных сертификатов** относится к типу 0x0 (список аннулированных сертификатов **системы управления платформой**).

4) Второй **сертификат** относится к типу 0x0 (**сертификат системы управления платформой**).

10.5 Специальные правила для цепочек экспорта/импорта

10.5.1 Обработка цепочки авторизации экспорта

Цепочка аутентификации экспорта и соответствующая секция сторонней цепочки экспорта передается в CPS.

CPS начинает обработку с минимальной корневой версии **списка аннулированных сертификатов** и версии, определенной в ss[slotId].se[sessionId].config.decryptConfig.minEciRootState. Она обрабатывает цепочку **сертификатов** EAOC и EAC и связанные с ними **списки аннулированных сертификатов**, проверяя соблюдение следующих специальных правил для этой цепочки:

- Идентификатор корневого RL равен 0x4 (авторизации экспорта по списку аннулированных сертификатов **оператора**).
- Идентификатор следующего **сертификата** (EAOC) равен 0x4.
- Идентификатор следующего **списка аннулированных сертификатов** (REAOC RL) равен 0x0.
- Идентификатор следующих **сертификатов** (EAC) в цепочке равен 0x0.
- Содержимое поля расширения **сертификата** равно соответствующему **сертификату** цепочки экспорта в цепочке экспорта.
- Идентификатор следующего **списка аннулированных сертификатов** (EAC-RL) в цепочке равен 0x0.
- Все **сертификаты** цепочки экспорта последовательно проверены по цепочке авторизации экспорта.
- Первым **сертификатом** сторонней секции цепочки экспорта является TPEGC (идентификатор сертификата равен 0x5).
- Последним **сертификатом** сторонней секции цепочки экспорта является TPEGC, ESC или ERC (идентификатор сертификата соответственно равен 0x5, 0xE или 0xF).
- Все промежуточные **сертификаты** являются EGC (идентификатор сертификата равен 0x4).
- Если последним **сертификатом** является TPEGC, он становится началом следующей секции цепочки экспорта. Вышеуказанный процесс проверки повторяется для всех последующих секций цепочек аутентификации экспорта и сторонних секций цепочки экспорта до тех пор, пока не будут получены полностью проверенные результаты сторонней цепочки экспорта (заканчивающиеся символом ESC или ERC).

10.5.2 Проверка цепочки экспорта

CPS начинает обработку с открытого ключа POC, индекса группы экспорта, для которой осуществляется экспорт, и минимального номера версии **списка аннулированных сертификатов**, который применяется к **списку аннулированных сертификатов** POC, находящемуся в поле состояния **сегмента AS** ss[slotId].se[sessionId].config.decryptConfig.minClientVersion.

ПРИМЕЧАНИЕ. – Такая проверка основана на надлежащей аутентификации POPK и версии **списка аннулированных сертификатов**. Она проводится с использованием либо аутентификации в режиме AK, либо неявной аутентификации с использованием **лестницы ключей** (см. пункт 8.2.2.2, поля klModeAuth и akModeAuth).

CPS обрабатывает POC-RL, EGC и EGC-RL и последующие TPEGC или ESC как обычную **цепочку сертификатов**. Проверяется выполнение следующих дополнительных правил:

- Тип EGC = 0x4.
- Поле EGC export_group_id равно индексу группы экспорта.
- Тип **списка аннулированных сертификатов** EGC = 0x4.
- Тип EGC-RL = 0x4.
- Тип TPEGC или ESC соответствует значению из таблицы 5.2-2 [ITU-T J.1012].

Процедура обработки TPEGC описана в пункте 10.5.3. Процедура обработки ESC описана в пункте 10.5.4.

10.5.3 Проверка сторонней цепочки экспорта

Обработка сторонней цепочки экспорта начинается с проверки начального сертификата TPEGC и минимального номера версии соответствующего **списка аннулированных сертификатов**. Обработка заканчивается сертификатом ESC.

10.5.4 Обработка сертификата системы экспорта

Для подтверждения **соединения экспорта** используется **сертификат** ESC SPK (открытый ключ ESC) и минимальный номер версии **списка аннулированных сертификатов родительского объекта** ESC. **Сертификат** SPK должен соответствовать полю ss[slotId].spk указанного сегмента экспорта. Минимальный номер версии **списка аннулированных сертификатов** должен превышать значение ss[slotId].ssConfig.microServerVersion для экспорта.

ПРИМЕЧАНИЕ. – Для обеспечения достоверной аутентификации сегмент экспорта SPK и параметр microServerVersion аутентифицируются **механизмом аутентификации** AK **сегмента AS**.

10.5.5 Правила обработки цепочки целевых клиентов

Обработка цепочки **целевых** клиентов начинается с POPK и минимального номера версии **списка аннулированных сертификатов** состояния **MSConfig микросервера**. Обработка цепочек **целевых** клиентов подсистемой CPS осуществляется по общим правилам, определенным в пункте 10.1. Кроме того, обработка цепочки **целевых** клиентов подсистемой CPS выполняется с соблюдением следующих специальных правил:

1) Первый список аннулированных сертификатов относится к типу 0x0 (список аннулированных сертификатов целевого клиента).
2) Первый сертификат относится к типу 0x0 (сертификат целевой группы) или 0x8 (сертификат микроклиента).
3) В том случае, если сертификат на этапе 2 является сертификатом целевой группы, этапы 1 и 2 повторяются.

Итоговый открытый ключ **сертификата микроклиента** представляет собой открытый ключ чипсета, который должен использоваться в соответствии с механизмом, описанным в пункте 7.3.

10.6 Инициализация корневого ключа ECI CPS

Во время инициализации **системы AS хост ECI** загружает в CPS последнюю информацию о применимом **корневом ключе ECI** и номере **списка аннулированных сертификатов**.

```
function InitCPSEciRoot(uchar minRootKeyVersion, uint minRevListNr)
```

Семантика

Выполняется следующий код на языке C:

```
cpsEciRootState.rootVersion = minRootKeyVersion;
cpsEciRootState.rlVersion   = minRevListNr;
```

CPS применяет значение **rootKeyVersion** в качестве номера версии **корневого ключа ECI** и значение **minRevListNr** ко всем цепочкам, предоставленным ей для загрузки учетных данных **ECI**.

Все остальные состояния **системы AS** сбрасываются.

Отметим, что настройка обоих параметров **хостом ECI** должна гарантировать, что все **клиенты ECI** могут быть загружены и что **хост ECI** не аннулирован, при этом ни у одного из **клиентов ECI** не отозван **сертификат**.

11 Ядро загрузчика

11.1 Введение

В системе **ECI** используется механизм загрузчика, который позволяет **клиентам ECI** надежно проверять версию загружаемых учетных данных **хоста ECI** и **клиента ECI** для выявления любой известной проблемы безопасности. Это позволяет выполнять обновление **хоста ECI** и **клиентов ECI** (как образов, так и ключей POPK) в качестве обычной функции работы системы.

Загрузчик образов **хоста ECI** и **клиентов ECI** использует определенные принципы **устойчивости**, оформленные в виде правил, которые описаны в следующих разделах. **Устойчивость** соблюдения этих правил определяется соответствующим документом, выходящим за рамки спецификации **ECI**, но в целом правила должны иметь равную **устойчивость** реализации. Считается, что некоторые правила реализуются с повышенной (высшей) **устойчивостью** и должны быть существенно более устойчивыми, чем реализация **хоста ECI**.

11.2 Правила загрузчика хоста

Загрузчик хоста ECI должен соблюдать следующие правила:

1) При инициализации после включения питания **загрузчик хоста ECI** должен обеспечить сохранение номера версии корневого ключа **ECI** и версии **списка аннулированных сертификатов** корневого ключа **ECI**, используемых для проверки образов **хоста ECI**, и после этого изменение этого номера должно стать невозможным. Это правило требует высшей **устойчивости**.
2) Должно быть невозможным изменение самого **загрузчика хоста ECI**. Это правило требует высшей **устойчивости**.
3) Должно быть невозможным изменение или считывание образа **хоста ECI** после его загрузки – в той мере, в какой это необходимо для предотвращения манипулирования конфиденциальной информацией или считывания секретной информации.
4) При любой последующей проверке образа **хоста ECI** (в случае поэтапного загрузчика), выполняемой программным обеспечением, взятым из предыдущего образа, должен использоваться один и тот же открытый ключ **ECI HostCertificate** и **список аннулированных сертификатов**.

Рекомендуется, чтобы поэтапные загрузчики использовали единый надежный механизм проверки образов **хоста ECI**, который также использовался для проверки первого загруженного образа **хоста ECI**.

11.3 Правила загрузчика клиента

Загрузчик клиента ECI зависит от **хоста ECI**. **Хост ECI** устанавливает минимальный номер версии корневого ключа **ECI** и **списка аннулированных сертификатов** корневого ключа **ECI**, которые он

использует для проверки **цепочек сертификатов** перед загрузкой любого связанного с клиентом элемента. **Загрузчик клиента ECI** должен соблюдать следующие правила:

1) Сначала образ **клиента ECI** дешифруется, если это требуется, как указано в пункте 11.5.

2) Образ **клиента ECI** и POPK проверяются с использованием цепочек, обработанных CPS, как определено в разделе 10. Это правило требует высшей **устойчивости**.

3) **Сертификат** образа **клиента ECI** или образа серии клиентов (в зависимости от ситуации) проверяется с помощью POPK и **списка аннулированных сертификатов системы управления платформой** клиента. Достоверность номера версии этого **списка аннулированных сертификатов** проверяется **клиентом ECI** позднее при инициализации сеанса **сегмента AS**.

4) Изменить или считать образ **клиента ECI** после его загрузки должно быть невозможно.

5) **Клиенты ECI** не должны иметь возможность "выйти за пределы своей программной среды" и наблюдать за поведением **хоста ECI** или **клиента ECI** или изменять его.

11.4 Обеспечение аннулирования сертификата

ECI использует устойчивый механизм обеспечения проверки учетных данных образов **хоста ECI** и **клиента ECI**. Он работает по следующим правилам:

1) Дескремблер прекращает работу в случае, если номер версии **корневого ключа ECI** и минимальный номер версии **списка аннулированных сертификатов** для проверки **цепочки сертификатов хоста ECI** оказываются ниже тех, что были загружены **хостом ECI** при инициализации. Это правило требует высшей **устойчивости**.

   ПРИМЕЧАНИЕ 1. – Это нетипичная ситуация, так как **список аннулированных сертификатов** корневого ключа **хоста ECI** регулярно обновляется по каналам всех **операторов**, и **загрузчик хоста ECI** может использовать последнюю версию этого **списка.**

2) **Система AS** отказывается загружать любого **клиента ECI**, **цепочка сертификатов** которого не может быть проверена с использованием версии корневого ключа **ECI** и минимального номера версии **списка аннулированных сертификатов ECI**, установленных **хостом ECI** при инициализации, как определено в пункте 11.2. Это правило требует высшей **устойчивости**.

3) **Сегмент AS** отказывается вычислять ключи в том случае, если минимальный номер версии корневого ключа и минимальный номер версии **списка аннулированных сертификатов**, требуемые **клиентом ECI**, оказываются ниже тех, что загружены **хостом ECI** при инициализации. Это определено в правилах вычисления ключей образа клиента, шифрования и дешифрования в пункте 8.2.4. Это правило требует высшей **устойчивости**.

   ПРИМЕЧАНИЕ 2. – Эти правила гарантируют, что перед выполнением любой связанной с защитой операции системы защиты контента могут потребовать применения минимального состояния корневого ключа **ECI** для проверки всех элементов, загруженных в **хост ECI**.

11.5 Дешифрование образа клиента

Для дешифрования образа **клиента ECI усовершенствованная система безопасности** может дешифровать предоставленный **оператором клиента ECI** ключ дешифрования зашифрованного образа, который дешифрует образ **клиента ECI**, как определено в пункте 7.8 [ITU-T J.1012]. Дешифрование образа **клиента ECI** выполняется до проверки подписи образа **клиента ECI**. Функцией AS, используемой для вычисления ключа дешифрования образа, является функция reqAsComputeImageKey, определенная в пункте 8.2.4.12. **Хост ECI** получает от **оператора** требуемую зашифрованную информацию о ключе inputV (входное сообщение в **механизм аутентификации**, по которому можно вычислить ключ аутентификации), ключ eKey (ключ дешифрования образа, зашифрованный ключом аутентификации) и параметры online и min_root_state, определенные в пункте 7.8 [ITU-T J.1012], и затем использует **сегмент AS**, предоставленный клиенту для выполнения дешифрования (см. пункт 7.8 [ITU-T J.1012]). Любой одноразовый код, используемый для сеанса обмена ключами дешифрования образа, должен быть новейшим (значение из заново инициализированного **сегмента AS**).

12 Требования синхронизации

12.1 Введение

Для того чтобы соответствовать требованиям системы безопасности, частью которой являются **клиенты ECI**, они должны функционировать с учетом определенных ограничений по времени. Поэтому **клиенты ECI** зависят от определенных характеристик функций, предлагаемых **системой AS** (через **хост ECI**). В этом разделе определены временны́е характеристики функций **системы AS**.

По временны́м характеристикам функции **системы AS** делятся на четыре категории:

1) Функции, которым требуются просто административные функции **в сегменте AS**.
2) Функции, которым требуются только симметричные криптографические операции, такие как вычисление **лестницы ключей** или дешифрование с помощью AK.
3) Функции, которым требуются от одной до четырех асимметричных криптографических операций в **блоке лестницы ключей** или CPS, таких как загрузка LK1 и выполнение функций, связанных с вычислением AK.
4) Функции, которым требуется обработка потенциально более длинных цепочек сертификатов, таких как цепочки импорта/экспорта и цепочки аутентификации **микроклиента**.

Функции трех последних категорий **клиент ECI** может вызывать посредством асинхронных сообщений. Функции первой категории могут быть синхронными или асинхронными.

Асимметричные криптографические операции занимают больше времени. Никакая выполняемая асимметричная криптографическая операция не должна блокировать функции первых двух категорий. Если для функций категории 1 или 2 требуется результат операции функции категории 3 или 4, то за синхронизацию результата функции категории 3 или 4 отвечает **клиент ECI**. Это означает, что прежде чем вызывать функцию, зависящую от результата, **клиент ECI** должен дождаться результата асимметричной операции (то есть получения сообщения с результатом).

12.2 Административные функции

К функциям категории 1 применяются общие критерии симметричных и асимметричных сообщений.

12.3 Симметричные криптографические функции

Функции, производящие асимметричные криптографические операции, должны выполняться **системой AS** так, чтобы каждый **сеанс сегмента AS** выполнял по одной функции за определенный период времени. Предлагаемое значение см. в пункте 6.6.3 [b-ITU-T J Suppl. 7].

12.4 Асимметричные криптографические функции

Функции, производящие асимметричные криптографические операции (например, с использованием вычислений симметричных ключей лестницы ключей или с использованием результата **механизма аутентификации**), должны выполняться **системой AS** так, чтобы каждый **сегмент AS** выполнял по одной функции за прием. Типичные значения предлагаются в пункте 6.6.4 [b-ITU-T J Suppl. 7].

Приложение A

Определения криптографических функций

(Данное Приложение является неотъемлемой частью настоящей Рекомендации)

A.1 Хеш-функция

Все хеш-функции, упоминаемые в настоящей Рекомендации, основаны на алгоритме SHA256, определенном в документе NIST FIPS PUB 180-4 [NIST FIPS 180-4].

Хеш-функция, упомянутая в пункте 7.2, эквивалентна функции SHA-256(), определенной в документе [NIST FIPS 180-4].

C-функция asHash (uchar *data, uint datalength, resultLength, uchar *result) использует октеты, начиная с данных длиной lengthLength в качестве октетной строки *dataIn*, и вычисляет октетную строку resultOut в виде октетной строки длиной length/8, сохраняя ее как результат в соответствии с уравнением:

resultOut = BS2OSP(truncate(SHA-256(OS2BSP(*dataIn*)),resultLength))).

Значение resultLength кратно 8. truncate – это функция усечения левой части битовой строки (параметр 1) до длины (параметр 2) бит.

BS2OSP и OS2BSP – это функции, преобразующие битовую строку в октетную строку и наоборот, как определено в разделе 9 [ITU-T J.1015].

A.2 Асимметричная криптография

Асимметричные операции шифрования и дешифрования определены в пунктах 10.2 и 10.3 [ITU-T J.1015].

A.3 Генерирование случайных чисел

Генерирование случайных чисел, как оно определено в настоящей Рекомендации, должно производиться в соответствии с требованиями [NIST 800-90Ar1] по следующим правилам:

- Как минимум при запуске системы (перезагрузка чипа **системы AS**) генерируется новое секретное уникальное случайное инициирующее число. Процесс зависит от физических (шум) или других свойств чипа или его среды, которые невозможно воспроизвести и которыми невозможно манипулировать. Энтропия сгенерированного числа должна быть не менее 128 бит.
- Любые случайные числа генерируются с помощью детерминированного генератора псевдослучайных чисел на основе указанного выше случайного инициирующего числа в соответствии с [NIST 800-90Ar1]. Чип может периодически обновлять инициирующее число в генераторе и/или увеличивать энтропию, как определено в пункте 8.7 [NIST 800-90Ar1], используя внутренние (шум) или внешние входные сигналы, которыми трудно манипулировать. В строке персонализации как минимум должен использоваться идентификатор чипа.

ПРИМЕЧАНИЕ. – Во многих применениях AS фактический уровень случайности генератора случайных чисел не критичен, а критична только уникальность генерируемых значений в каждый момент времени. Это типичные применения одноразового кода: например, случайное число для онлайн-аутентификации для предотвращения повторного воспроизведения при дешифровании и вставка случайного числа при шифровании контента. Исключение составляет случайный ключ, сгенерированный как LK1 в **сегменте AS** шифрования в асимметричном режиме **микросервера**.

Дополнение I

Пример применения системы Micro DRM

(Данное Дополнение не является неотъемлемой частью настоящей Рекомендации)

I.1 Введение

В этом дополнении приводится реальный пример применения **системы AS** для реализации **системы Micro DRM**, работающей в потоке TS. Представлен пример операций как шифрования, так и дешифрования **клиентов ECI**. Основное внимание уделяется параллельному выполнению различных действий и последовательности управляющих слов и связанных с ними сообщений Micro DRM (от **микросервера** к **микроклиенту** и наоборот), которые необходимо создать. **Система Micro DRM** создает как случайный ключ при шифровании, так и одноразовый код при дешифровании (для предотвращения повторного воспроизведения). Предполагается, что оба случайных ключа ограниченны.

I.2 Сценарий применения

В сценарии применения на рисунке I.1 показано состояние **лестницы ключей** со стороны шифрования. LKn – это третий, но самый младший ключ в иерархии ключей. Ниже расположены одноразовые коды (N1 или N2) из **микроклиента**, **свойства контента** CP1 и CP2 (преобразуемые во входные данные на языке C для **лестницы ключей** на шаге n+2) и инициирующие числа случайных ключей R1 и R2, которые вводятся в **лестницу ключей** на шаге n+3. По этим входным данным **лестницы ключей** вычисляются управляющие слова CW1..CW4 и применяются к контенту в сочетании с соответствующими **свойствами контента**.

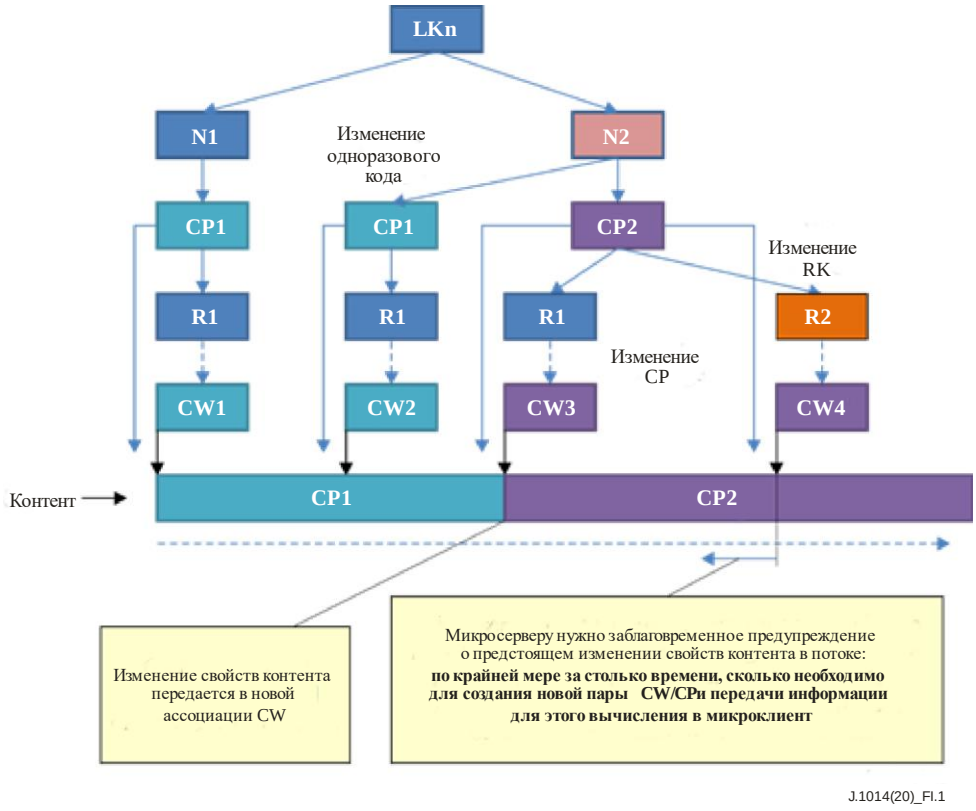

Рисунок I.1 – Пример эволюции иерархии ключей для вычисления управляющих слов

Начальным состоянием нижних трех ступеней иерархии ключей **микросервера** является состояние N1, CP1 и R1. По этим данным вычисляется CW1 для шифрования контента. Исходным состоянием бита переключения является t1. В этом примере **микросервер** сначала получает новый одноразовый код и решает применить его к будущему контенту в виде CW2. Сначала он отправляет **микроклиенту**

ECM-сообщение с новым битом переключения t2 и набором зашифрованных ключей (t2, N2, CP1, R1) и ожидает некоторое время, чтобы убедиться, что **микроклиент** имеет возможность получить и предварительно вычислить новое управляющее слово CW2 и подготовиться к предстоящим изменениям. Затем он сам вычисляет новое управляющее слово CW2 и переходит к его применению, вызывая изменение в бите переключения соответствующего зашифрованного потока TS.

Следующее событие на рисунке I.1 показывает изменение **свойств контента** шифруемого контента. **Микросервер** получает от **хоста ECI** сообщение о том, что **свойства контента** будут изменены на CP2. **Микросервер** отправляет своему **микроклиенту** ECM-сообщение с новым набором зашифрованных ключей (t3, N2, CP2, R1) и по этой новой последовательности ключей предварительно вычисляет CW3. В этот момент применяются новые **свойства контента**, бит переключения в зашифрованном контенте автоматически изменяется, и фактически применяются новое управляющее слово шифрования CW3 и связанные с ним **свойства контента** CP2.

Последнее событие – это решение **микросервера** изменить случайный ключ R1 на R2. Процесс практически идентичен процессу изменения одноразового кода. **Микросервер** отправляет **микроклиенту** ECM-сообщение, содержащее (t4, N2, CP2, R2) и разрешающее ему предварительно вычислить CW4, и использует задержку, чтобы гарантировать **микроклиенту** достаточное время для подготовки. Затем он использует **лестницу ключей** для вычисления CW4 и применяет его к контенту, так что бит переключения контента устанавливается в состояние t4.

I.3 Предположения и условные обозначения

Контент доставляется экспортирующим клиентом и связанным с ним **сегментом AS** дешифрования в соединение импорта **сегмента AS** шифрования. Экспортирующий клиент направляет сообщения **хосту ECI**, сигнализируя о любых изменениях **свойств контента** до их фактического наступления в импортируемом контенте. При этом используется API AS из [ITU-T J.1012].

Применяются следующие условные обозначения:

<event-name>(parameters) -> <pseudo-code statement>; указывает, что по событию event-name (прием сообщения) выполняется следующий псевдокод.

Определены следующие события:

- **e_cp(cp):** в предстоящем событии (изменение CW) в зашифрованном контенте будут использоваться новые свойства контента cp. Предшествует `e_cpe()`.
- **e_cpe():** значимое изменение свойств контента (должно уложиться в ограниченное время).
- **e_cpch():** свойства импортированного контента только что изменились, и если используемое в настоящее время управляющее слово не отражает этого, его требуется срочное заменить. При автоматическом изменении управляющего слова ввиду изменения свойства управления это событие предшествует событию `e_cw()`.
- **e_nn(nonce):** новое сообщение с одноразовым кодом от **микроклиента** поступило в **микросервер** или отправлено из **микроклиента**.
- **e_cw():** в перешифрованном контенте изменился бит переключения, и к этому контенту применяется новое (заранее вычисленное) CW.
- **e_ecm(<parameters>):** прием сообщения с новыми параметрами следующего управляющего слова, которое будет использоваться.
- События могут запускаться по таймерам.

cw(toggle_bit,random_key,nonce,content_properties) выполняет генерирование управляющего слова для шифрования или дешифрования контента с использованием назначенных параметров. В **микросервере** сначала генерируется сообщение с теми же параметрами, которое передается в **микроклиент** и там принимается как `e_ecm(…)`.

block_cpch() и unblock_cpch() используют сообщение setAsPermitCPChange(...), чтобы заблокировать или разблокировать автоматические изменения в управляющем слове шифрования вследствие изменений в **свойствах контента** импортируемого контента.

changeCw(toggleBit) вызывает замену управляющего слова (бит переключения в поле управления скремблированием) со стороны шифрования с использованием сообщения setAsSC(), как указано в пункте 9.9.

startTimer(timerHandle) запускает таймер.

Для переменных и псевдокода используется запись в стиле языка C.

I.4 Псевдокод микросервера

Усложнение ключа в **микросервере** означает, что он должен обрабатывать несколько одновременных и несинхронизированных событий, которые могут вызвать изменение управляющего слова:

- прибытие контента, требующего новых свойств контента (что характеризуется применением экспортирующим клиентом нового управляющего слова к дешифровываемому контенту);
- предстоящее истечение срока действия одноразового кода; а также
- предстоящее истечение срока действия случайного ключа.

Обработка изменений в **свойствах контента** должна иметь приоритет, поскольку, как правило, имеется лишь ограниченное время до того, как изменение соответствующего управляющего слова в процессе дешифрования инициирует применение новых свойств контента. Таким образом, сроки действия одноразового кода и случайного ключа должны быть установлены достаточно консервативно, поскольку их, возможно, придется задержать на время, необходимое для обработки изменения свойств контента (обычно на несколько секунд). Это предполагает, что между изменениями свойств контента всегда достаточно времени для необходимой обработки по меньшей мере с целью замены одноразового кода или случайного ключа в управляющем слове.

Обработка значимых изменений одноразового кода или случайного ключа имеет два уровня приоритета. Сначала устанавливается таймер для низкоприоритетных изменений. Если никаких изменений в **свойствах контента** не ожидается, то выполняется изменение одноразового кода или случайного ключа, в противном случае устанавливается таймер для высокоприоритетного изменения. Высокоприоритетное изменение одноразового кода или случайного ключа может отменить значимое изменение свойств контента. Однако **микросервер** может прийти к неверному заключению. Это возможно в том случае, если изменение свойства контента произошло до применения к контенту изменившегося одноразового кода или случайного ключа. Тогда требуется заново вычислить управляющее слово, которое также включает новые **свойства контента**. Кроме того, изменение свойств контента может произойти почти сразу же после применения высокоприоритетного изменения одноразового кода или случайного ключа. В этом случае CW, отражающее новые **свойства контента**, и вычисляемое микросервером ECM будут запоздалыми.

Такой конфликт может произойти, если значения таймеров TNONCEURGENT и TRKURGENT можно установить на время, превышающее 10 секунд плюс TECM, а максимальное время между e_cpch () и e_cw(CPCHANGE) составляет меньше 10 секунд, поскольку любое изменение RK и одноразового кода может быть запланировано либо до периода между e_cpch () и e_cw(CPCHANGE), либо после такого периода без установки необходимого приоритета.

Отметим, что манипулирование переменными rc и rn, как указано ниже, не может выполняться непосредственно клиентом, а только с использованием функций **системы AS**.

```
/*
   четырехприоритетная модель обработки с небольшим сдвигом времени изменения CP
   в случае необходимости приоритета 4 (непредвиденное изменение CP здесь и сейчас):
    1) низкоприоритетное изменение одноразового кода/rk
    2) низкоприоритетное изменение CP (cp значимо, но событие e_cpch() не происходит)
       учитывает любые предыдущие изменения одноразового кода или rk
    3) высокоприоритетное изменение одноразового кода или rk; возврат к прежнему значению CP
    4) высокоприоритетное изменение CP; учитывает ожидающие изменения одноразового кода/rk
и новые cp;
       новые изменения ставятся в очередь.

    Возможна оптимизация в виде попытки планировать предстоящие изменения одноразового кода и rk
    сразу после изменения CP; обеспечивает умеренное повышение производительности

    Инварианты/значения переменной состояния:
    <x> = cp (свойство контента), n (одноразовый код) или r (случайный ключ)
    Инвариант: p<x> = изменение <x> в следующем CW (p = ожидающий)
                      (не для низкоприоритетных <n> и <r>)
```

```
              Q <x> = очередное изменение <x>, не ожидая следующего CW
              hpcp = высокоприоритетное изменение свойства контента (pcp || qcp)
   В течение короткого промежутка времени между changeCw() и e_cw() все изменения ставятся в
очередь.
   Это временное состояние указывается с помощью выражения dhp==true;
 */

#define TECM 3000 /* задержка между отправкой ecm-сообщения и изменением CW */
#define TNONCEURGENT (2*TEMC + 1000)
#define TRKURGENT (2*TECM + 1000)
#define TNONCE /* некоторое значение; может определяться динамически*/
#define TRK    /* некоторое значение; может определяться динамически*/

toggle(bool t) { return !t }; /* переключение между значениями "истинно" и "ложно" */

encryptionSession()
/* случай изменения rk и одноразового кода и изменения cp; ненадежное предупреждение об изменении cp
(изменение приоритета одноразового кода/RK) */
/* сначала приоритет одноразового кода/rk ниже, чем у cp, но в случае срочности он выше */
{
  SymKey  nc, nn;   /* текущий и следующий одноразовый код */
  SymKey  rc, rn;   /* текущий и следующий случайный ключ */
  SymKey  cpc, cpn; /* текущее и следующее значение CP */
  SymKey  nt, rt;   /* временное значение одноразового кода или случайного ключа */
  TimerHandle t_lpn, t_lpr;
              /* таймеры для планирования низкоприоритетных изменений одноразового кода и
случайного ключа */
  TimerHandle t_n, t_r;
              /* таймеры для планирования высокоприоритетных изменений одноразового кода и
случайного ключа */
  TimerHandle t_ecm_n1, t_ecm_r1;
              /* таймеры ecm низкоприоритетных ecm (1) одноразового кода и случайного ключа */
  Timerhandle t_ecm0, t_ecm1 t_ecm2, t_ecm3;
  TimerHandle t_ecm[4] = {t_ecm0, t_ecm1, t_ecm2, t_ecm3 };
              /* четырехуровневый пул таймеров приоритета ecm 2/3/4 */
  int t_ecm_cnt = 0;  /* счетчик распределения вышеуказанного пула таймеров */
  bool pn, pr, pcp;   /* истинно, если текущее CW отражает изменение значения одноразового кода
(nn),
                         случайного ключа (rn) или cp (cpn) */
  bool qn, qr, qcp; /* истинно в случае изменения находящихся в очереди изменений одноразового кода,
случайного ключа или cp */
  bool dhp;         /* задержка (очередь) любых новых событий */
  book hpcp;        /* истинно, если приоритет 4: высокоприоритетное изменение CP */
  int  tCnt1, tCnt234;  /* tCnt <n> - это счетчик числа таймеров
                           приоритета <n>, которые запущены, но еще не сработали */
  bool t;           /* бит переключения */

/* определены некоторые макросы для повторного использования кода событий обработки */

.* событие следующего случайного ключа */
#define next_r() { rc = rn; rn = rnd128(); startTimer(t_lpr,TRK); }


/* вызывает changeCw по последнему в цепочке высокоприоритетному таймеру, если это не
   изменение приоритета cp уровня 2 - в этом случае изменение CW инициируется
   событием изменения CP */
#define process_emc2_timer(){\
  if (--tCnt234 == 0)\
    if (pn || pr || hpcp){\
      dhp = true; changeCw(toggle(t));\
    } else {\
      /* pcp == true, pn, pr, hpcp == false */\
      unblock_cpch();\
    };\
}

/* при изменении cw обновляется состояние с учетом всех обработанных изменений */
#define end_pending() {\
  t = toggle(t);\
  if (pcp) { cpc = cpn; pcp = false };\
  if (pn)  { nc = nn; pn = false };\
  if (pr)  { next_r(); pr = false };\
}

/* перенос событий из очереди в состояние ожидающих */
#define queued_to_pending() {\
  if (qcp && (!(qn || qr) || cphp)) {
    /* если приоритет 2 или 4 */\
    pcp = true; qcp = false\
  };\
  /* события приоритета 3 могут быть охвачены приоритетом 4 */
  if (qn)  { pn = true; qn = false };\
  if (qr)  { pr = true; qr = false };\
```

```
}

/* запуск cw/ecm в ожидании изменений в cw */
#define start_pending() {\
  cnt = 0;\
  if (pcp) { cpt = cpn; cnt++ } else cpt = cpc;\
  if (pn)  { nt  = nn ; cnt++ } else nt  = nc;\
  if (pr)  { rt  = rn ; cnt++ } else rt  = rc;\
  if (cnt > 0) {\
    block_cpch();\
    cw(t,rt,nt,cpt);\
    tCnt234++;\
    startTimer(t_ecm[t_ecm_cnt++],TECM);\
    if (t_ecm_cnt >=4) t_ecm_cnt = 0 ;\
  }\
}


  /* разрешены только подготовленные автоматические изменения бита переключения */
  block_cpch();

  /* получение первых значений cp и одноразового кода */
  for (int i=0; i<2;) {
    ->e_nn(&nc): i++;
    ->e_cp(&cpc): i++;
  }

  /* инициализация состояния */
  pn = pr = pcp = hpcp = false;
  dhp = false;
  tCnt1 = tCnt2 = 0;
  rc = rnd128(); rn = rnd128() ;
  t = false;          /* должно быть инициализировано в первое значение контента */
  cw(t,rc,nc,cpc) ;   /* начинает использоваться автоматически */

  while (!end_session) {
  ->e_nn(&nn) : startTimer(t_lpn,TNONCE) ;
                /* должно произойти до истечения срока действия одноразового кода */
  ->e_cp(&cpn): /* например, вычисление новых экспортных лицензий */ ;
  ->t_lpn()   : { /* низкоприоритетное изменение одноразового кода */
      if (pcp || pn || pr || cphp) {
         /* задержка нового одноразового кода до того, как это станет срочно необходимым */
         startTimer(t_n,TNONCEURGENT);
      } else {
        nc = nn;
        cw(t,rc,nc,cpc);
        startTimer(t_ecm_n1,TECM) ;
        tCnt1++;
      }
    };
  ->t_lpr()   : { /* низкоприоритетное изменение rk */
      if (pcp || pn || pr || cphp) {
        /* задержка РК до того, как это станет срочно необходимым */
        startTimer(t_r,TRKURGENT);
      } else {
        next_r();
        cw(t,rc,nc,cpc);
        startTimer(t_emc_r1,TEMC);
        tCnt1++;
      }
    };
  ->t_emc_n1() :   /* истечение срока действия низкоприоритетного таймера ecm одноразового кода */
  ->t_emc_r1() : { /* истечение срока действия низкоприоритетного таймера ecm rk */
      if (--tCnt1 == 0 && tCnt234 == 0) {
        changeCw();
        dhp = true;
      }
    };
  ->e_cpe()   : { /* с этого момента может произойти изменение cp */
      if (dhp || (pn || qn)) qcp = true; /* assert(!hpcp) */
      else { pcp = true; start_pending() };
    };
  ->t_n()      : { /* необходимо срочное изменение одноразового кода */
      if (dhp || cphp) qn = true;
      else { pn = true; start_pending() };
    };
  ->t_r()      : { /* необходимо срочное изменение случайного ключа */
      if (dhp || cphp) qr = true;
      else { pr = true; start_pending() };
    };
  ->e_cpch()   : { /* требуется высокоприоритетное изменение CP */
```

```
      cphp = true;
      if (dhp) qcp = true;
      else {
        pcp = true;
        start_pending();
      }
    };
  ->t_ecm0()   :
  ->t_ecm1()   :
  ->t_ecm2()   :
  ->t_emc3()   : {
      process_timer();
    };
  ->e_cw()     : { /* assert( pcp && !pn && !pr && !cphp ) */
      end_pending();
      queued_to_pending();
      start_pending();
    };
  }
}
```

ПРИМЕЧАНИЕ 1. – В представленном выше примере **микросервер** может генерировать, а **микроклиент** – принимать несколько последовательных, но разных сообщений ECM с одним и тем же битом переключения. Крайним случаем является пример, когда сначала происходит `e_n()`, затем в течение времени `TDELAY` происходит `e_r()`, а затем в течение времени `TDELAY` происходит `e_cp(..)`. В этом случае отправляются три последовательных сообщения ECM с одним и тем же битом переключения, причем только последнее сообщение приводит к управляющему слову, которое фактически применяется к контенту.

ПРИМЕЧАНИЕ 2. – В приведенном выше коде предполагается, что за изменением свойств контента `e_cp()` всегда относительно быстро следует изменение фактического бита переключения. Если новое значение cp будет доступно значительно раньше, это не принесет **микросерверу** никакой пользы. Ключевой начальной точкой генерирования нового CW служит событие, при котором изменение cp во входящем контенте является значимым. Это приводит к тому, что во всех предстоящих вычислениях CW старое значение `cp` заменяется новым.

Минимальное время предварительного предупреждения для запуска `e_n()` или `e_r()` в приведенном выше примере кода представляет собой наихудший случай задержки между событием `e_cp()` и фактическим изменением последующего управляющего слова `e_cw()` плюс 2 × `TDELAY` плюс незначительная задержка и время обработки.

I.5 Псевдокод микроклиента

Сеанс **микроклиента** начинается с генерирования двух последовательных сообщений с одноразовым кодом (текущим и следующим). Если он получает сообщение ECM, то просто вычисляет соответствующее управляющее слово. Определив, что последнее отправленное сообщение применено в ECM, он продолжает генерировать новые сообщения с одноразовым кодом и отправлять их.

ПРИМЕЧАНИЕ 1. – Безопасные одноразовые коды нельзя генерировать непосредственно в коде **клиента ECI**; для этого должна использоваться соответствующая функция **системы AS**.

```
decryptionSession()
{
  Symkey nc, nn, ln;   /* текущий, следующий и последний одноразовый код */
  SymKey cp, cpp;  /*  принятое и предыдущее cp */
  SymKey r;        /* принятый случайный ключ */
  bool   t;        /* принятый бит переключения */
  SymKey n;        /* принятый одноразовый код */
  bool   end_session; /* достигнут конец сеанса */


  /* инициализация и отправка одноразовых кодов */
  nn = rnd128();
  e_nn(nn);
  cpp = Reserved; /* неопределенное значение */
  ln  = Reserved;

  while (!end_session) {
    ->e_ecm(&t,&r&n&cp): {
        if (cp!=cpp) { /* новый CP; событие отправляется по всем экспортным соединениям через
хост */
          e_cp(cp);
          cpp = cp;
        }
        cw(t,r,n,cp);
      };
    ->e_cw(): {         /* также срабатывает при первом применении cw */
```

```
        if (n != ln) {   /* новый фактически используемый одноразовый код; переход к следующему
одноразовому коду */
          nc= nn; nn= rnd128();
          e_nn(nn);
          ln = n;
        }
      };
  } /* конец цикла */
} /* конец сеанса дешифрования */
```

ПРИМЕЧАНИЕ 2. – Нет необходимости возвращать полные значения одноразового кода из **микросервера** в **микроклиент**. Вместо этого в качестве косвенной ссылки можно использовать бит переключения. Кроме того, не обязательно отправлять все параметры во всех ECM: **микроклиенту** должны передаваться только изменения с учетом того, что в некоторых случаях все три входа **лестницы ключей** – одноразовый код, **свойства контента** и случайный ключ – могут измениться одновременно (см. Примечание 2 в пункте I.4). Отправка бита переключения также полезна для синхронизации и позволяет избежать того, что любое (намеренное или непреднамеренное) повторное сообщение ECM интерпретируется как сообщение для вычисления следующего управляющего слова.

I.6 Влияние каскадирования системы Micro DRM на предзадержку ECM

Для сервера **Micro DRM** большое значение имеет период заблаговременного предупреждения (предзадержки) о предстоящем изменении свойств контента от **микроклиента**, из которого он импортирует контент, чтобы предварительно вычислить сообщение ECM и отправить его своему **микроклиенту**. Время, необходимое для выполнения требуемой обработки (которое может быть относительно коротким: значительных вычислений обычно не требуется), плюс время пересылки этого сообщения ECM в **микроклиент** может превышать время передачи вновь зашифрованного контента в **микроклиент**. Это означает, что любая задержка перед отправкой нового ECM, которую испытывает **микроклиент**, соответственно короче времени задержки, которую испытывает **клиент ECI**, из которого импортирован контент.

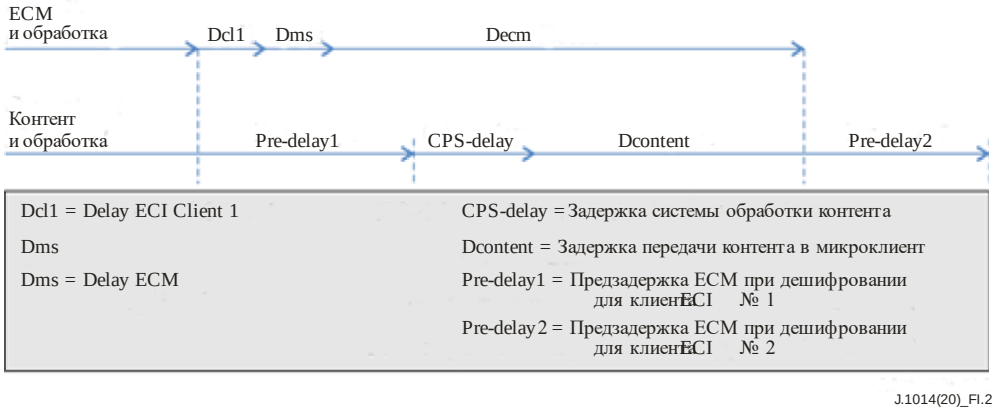

Рисунок I.2 – Временны́е отношения для компенсации предзадержки и дополнительной задержки

Безопасный видеотракт может ввести задержку при передаче контента для компенсации задержки при пересылке сообщений ECM, как показано на рисунке I.2. Эту задержку можно выбрать примерно равной разности задержек. В случае ECM вводимая задержка Decm и задержка Dcontent практически совпадают. Однако следует компенсировать задержки обработки в дешифрирующем **клиенте ECI** и в **микросервере** до момента фактического ввода ECM в поток TS.

I.7 Соглашение об интерфейсе синхронизации изменений свойств контента

Как показано в пункте I.4, **микросерверу** требуется предварительное предупреждение о предстоящем изменении свойств контента в контенте, который он импортирует. Соглашение о минимальном периоде времени, необходимом для обработки изменения и отправки сообщения ECM в **микроклиент**, обозначается как TECM: в нижеследующих пунктах этого дополнения приведен пример. Для этого примера установлено значение TECM, равное 3 с.

Соглашение о минимальной задержке предварительного предупреждения первого **клиента ECI** дешифрования в цепочке каскадированных **клиентов ECI** представляется как TECM + TCASCADE. TCASCADE отражает максимальную совокупную задержку обработки сообщений ECM **клиентами ECI** в каскадной цепочке **систем Micro DRM**. В этом примере TCASCADE равно 2 с.

ПРИМЕЧАНИЕ 1. – В тех случаях, когда контент также задерживается, это компенсирует задержки обработки ECM. Однако в потоковом режиме это нежелательно.

Максимальное значение задержки обработки ECM **клиента ECI** (некомпенсированная Dcl1 + Dms, как в пункте I.6) обозначается как TDELAY и в этом примере равно 0,3 с. Значения в этом примере позволяют использовать шесть каскадированных **систем Micro DRM** в пределах TCASCADE (2 с), оставляя минимальный период предварительного предупреждения TECM для **микросервера**.

Как показано в пункте I.4, чтобы обработать изменение свойств контента во входящем контенте, не вызывая сдвига изменения свойств контента, **микросерверу** требуется не только предварительное предупреждение о предстоящем изменении контента, но и значение верхнего предела такого периода предварительного предупреждения, чтобы он мог безопасно обрабатывать другие изменения управляющего слова (например, изменения одноразового кода и случайного ключа). В этом примере верхний предел периода предварительного предупреждения TMAXWARN можно безопасно установить равным 10 с.

ПРИМЕЧАНИЕ 2. – В том случае, если эти соглашения не соблюдаются, результатом может быть сдвиг, не превышающий TECM, в точке изменения свойства контента в повторно шифруемом контенте в одной **системе Micro DRM** и сдвиг, не превышающий TECM+6*TCASCADE, в каскаде из шести **систем Micro DRM**.

Настоятельно рекомендуется достаточно рано разработать низкоприоритетные предупреждения по одноразовому коду и случайному ключу (t_lpn и t_lpr в пункте B.4), чтобы разрешить одно (или даже несколько) изменение **свойств контента** для задержки обработки изменений одноразового кода и случайного ключа. Если изменения свойств контента достаточно разнесены по времени (и соблюдается TMAXWARN), это предотвратит любые накладки при обработке изменений одноразового кода и случайного ключа.

Выбор временны́х параметров важен для беспрепятственной передачи контента между **клиентами ECI**. Дополнительные сведения о рекомендуемых значениях параметров задержки TECM, TCASCADE, TDELAY и TMAXWARN приведены в разделе 6 [b-ITU-T J Suppl. 7].

Дополнение II

Тематические области, требующие доработки

(Данное Дополнение не является неотъемлемой частью настоящей Рекомендации)

Определено, что настоящая Рекомендация нуждается в доработке и валидации, с тем чтобы обеспечить ее соответствие требованиям, установленным в [ITU-T J.1010], и что необходимо обновить Рекомендацию [ITU-T J.1010], отразив в ней требования спецификации системы расширенной защиты контента MovieLabs (ECP) [b-ECP]. Рекомендации [ITU-T J.1011], [ITU-T J.1012], [ITU-T J.1013], МСЭ-Т J.1014, [ITU-T J.1015] и [b-ITU-T J.1015.1] следует в дальнейшем обновить, отразив в них указанные изменения в [ITU-T J.1010].

Ряд Государств – Членов МСЭ наряду с заинтересованными сторонами со всего мира, представляющими самые разные отрасли, включая производителей устройств и электронных компонентов, владельцев и лицензиатов авторских прав на контент, поставщиков услуг на базе технологии over-the-top (OTT) и линейного телевидения, а также поставщиков решений для систем условного доступа (CAS) и управления цифровыми правами (DRM), выразили обеспокоенность тем, что встроенный общий интерфейс (ECI) не в полной мере отвечает требованиям ECP и требованиям к защите контента, предъявляемым в более широком круге отраслей.

Соответствующие вопросы были, в частности, подняты во вкладах к собранию 9-й Исследовательской комиссии МСЭ-Т (ИК9) (16–23 апреля 2020 года). Во вкладах, представленных Израилем, Австралией, Членом Сектора МСЭ-Т компанией Samsung, а также Ассоциированными членами ИК9 компаниями Sky Group и MovieLabs, предлагалось внести ряд изменений в Рекомендации по тематике ECI, однако согласие по ним достигнуто не было. Эти предложения перечислены в [b-SG9 Report 17 Ann.1].

Они состояли, в частности, в следующем:

1) упростить систему ECI, ограничив сферу ее применения;
2) отказаться от DRM;
3) отказаться от повторного шифрования контента;
4) отказаться от управления программным обеспечением;
5) добавить интерфейсы API для защищенного хранения и криптографических операций;
6) предусмотреть возможность использования лестниц ключей, определяемых поставщиком;
7) установить требования к TEE, изложенные в Рекомендации J.1207;
8) включить в Рекомендации реализацию TEE для виртуальной машины;
9) применять более стойкие алгоритмы шифрования, например SHA-384;
10) использовать стандартные сертификаты, подобные приведенным в Рекомендации МСЭ-Т X.509;
11) пересмотреть обмен данными между клиентами;
12) организовать дополнительное взаимодействие с ETCИ;
13) провести дополнительное коллегиальное рассмотрение;
14) рассмотреть возможные альтернативы модели доверительного органа;
15) уточнить технические аспекты правил соответствия и обеспечения устойчивости ECI;
16) добавить требования об обеспечении технического разнообразия, например о рандомизации распределения адресного пространства;
17) добавить требования о проверке целостности данных на этапе выполнения.

Эти предложения отражают постоянную эволюцию защиты контента и способов ее нарушения. Первоначальный замысел ECI возник почти за десять лет до утверждения настоящей Рекомендации МСЭ-Т. Системы, подобные ECI, необходимо регулярно оценивать на предмет стойкости к современным методам осуществления атак, а также соответствия отраслевым требованиям к защите.

Существуют и другие механизмы обеспечения функциональной совместимости. В частности, что касается применения DRM, большинство интернет-служб доставки видео внедрили другие решения, обеспечивающие функциональную совместимость наряду с решением стоящих перед этими службами задач.

Важной задачей является внесение большей ясности, так как многие Государства-Члены рассматривают стандарты МСЭ как авторитетные источники руководящих указаний по развитию их рынков и отраслей. Упомянутый выше список предложений призван способствовать тому, чтобы можно было в полной мере уяснить все последствия, связанные с настоящей Рекомендацией МСЭ-Т, при внедрении ECI на рынках этих государств и учесть все возможные вопросы при рассмотрении законодательных и нормативных актов и потребностей рынка, требующих обеспечения функциональной совместимости потребительского цифрового телевизионного оборудования. Он также позволяет производителям оборудования, предпочитающим брать за основу при проектировании особые наборы требований или иные стандарты, учитывать эти вопросы в процессе разработки продукции, предназначенной для различных рынков.

Библиография

[b-ITU-T J.1015.1] Recommendation ITU-T J.1015.1 (2020), *Embedded common interface for exchangeable CA/DRM solutions; Advanced security system - Key ladder block: Authentication of control word-usage rules information and associated data 1*.

[b-ITU-T J Suppl. 7] Supplement 7 to the ITU-T J series Recommendation (2020), *Embedded Common Interface for exchangeable CA/DRM solutions; Guidelines for the implementation of ECI*.

[b-SG9 Report 17 Ann.1] ITU-T SG9 meeting report, SG9-R17-Annex 1 (2020), Annex 1 to Report 17 of the SG9 fully virtual meeting held 16-23 April 2020. https://www.itu.int/md/T17-SG09-R-0017/en

[b-ETSI GS ECI 001-1] ETSI GS ECI 001-1 V1.2.1 (2018), *Embedded Common Interface (ECI) for exchangeable CA/DRM solutions; Part 1: Architecture, Definitions and Overview*. https://www.etsi.org/deliver/etsi_gs/ECI/001_099/.../gs_ECI00101v010101p.pdf

[b-ETSI GS ECI 001-2] ETSI GS ECI 001-2 V1.2.1 (2018), *Embedded Common Interface (ECI) for exchangeable CA/DRM solutions; Part 2: Use cases and requirements*. https://www.etsi.org/deliver/etsi_gs/ECI/001_099/.../gs_ECI00102v010201p.pdf

[b-ETSI GS ECI 001-3] ETSI GS ECI 001-3 V1.1.1 (2017), *Embedded Common Interface (ECI) for exchangeable CA/DRM solutions; Part 3: CA/DRM Container, Loader, Interfaces, Revocation*. https://www.etsi.org/deliver/etsi_gs/ECI/001_099/.../01.../gs_ECI00103v010101p.pdf

[b-ETSI GS ECI 001-4] ETSI GS ECI 001-4 V1.1.1 (2017), *Embedded Common Interface (ECI) for exchangeable CA/DRM solutions; Part 4: The Virtual Machine*. https://www.etsi.org/deliver/etsi_gs/ECI/001_099/.../gs_ECI00104v010101p.pdf

[b-ETSI GS ECI 001-5-1] ETSI GS ECI 001-5-1 V1.1.1 (2017-07), Embedded *Common Interface (ECI) for exchangeable CA/DRM solutions; Part 5: The Advanced Security System; Sub-part 1: ECI specific functionalities*. https://www.etsi.org/deliver/etsi_gs/ECI/001_099/0010501/01.01.01_60/gs_ECI0010501v010101p.pdf

[b-ETSI GS ECI 001-5-2] ETSI GS ECI 001-5-2 V1.1.1 (2017), *Embedded Common Interface (ECI) for exchangeable CA/DRM solutions; Part 5: The Advanced Security System; Sub-part 2: Key Ladder Block*. https://www.etsi.org/deliver/etsi_gs/ECI/001_099/.../gs_ECI0010502v010101p.pdf

[b-ETSI DVB CSA] ETSI: *Using the DVB CSA algorithm* (лицензионное соглашение). http://www.etsi.org/about/what-we-do/security-algorithms-and-codes/csa-licences

[b-ETSI DVB CSA3] ETSI: *Using the DVB CSA3 algorithm* (условия лицензирования). http://www.etsi.org/about/what-we-do/security-algorithms-and-codes/csa3-licences

[b-ECP] MovieLabs Specification for Enhanced Content Protection – Version 1.2. Доступно по адресу: https://movielabs.com/ngvideo/MovieLabs_ECP_Spec_v1.2.pdf

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ-Т

Серия A	Организация работы МСЭ-Т
Серия D	Принципы тарификации и учета и экономические и стратегические вопросы международной электросвязи/ИКТ
Серия E	Общая эксплуатация сети, телефонная служба, функционирование служб и человеческие факторы
Серия F	Нетелефонные службы электросвязи
Серия G	Системы и среда передачи, цифровые системы и сети
Серия H	Аудиовизуальные и мультимедийные системы
Серия I	Цифровая сеть с интеграцией служб
Серия J	**Кабельные сети и передача сигналов телевизионных и звуковых программ и других мультимедийных сигналов**
Серия K	Защита от помех
Серия L	Окружающая среда и ИКТ, изменение климата, электронные отходы, энергоэффективность; конструкция, прокладка и защита кабелей и других элементов линейно-кабельных сооружений
Серия M	Управление электросвязью, включая СУЭ и техническое обслуживание сетей
Серия N	Техническое обслуживание: международные каналы передачи звуковых и телевизионных программ
Серия O	Требования к измерительной аппаратуре
Серия P	Качество телефонной передачи, телефонные установки, сети местных линий
Серия Q	Коммутация и сигнализация, а также соответствующие измерения и испытания
Серия R	Телеграфная передача
Серия S	Оконечное оборудование для телеграфных служб
Серия T	Оконечное оборудование для телематических служб
Серия U	Телеграфная коммутация
Серия V	Передача данных по телефонной сети
Серия X	Сети передачи данных, взаимосвязь открытых систем и безопасность
Серия Y	Глобальная информационная инфраструктура, аспекты межсетевого протокола, сети последующих поколений, интернет вещей и "умные" города
Серия Z	Языки и общие аспекты программного обеспечения для систем электросвязи

Отпечатано в Швейцарии
Женева, 2021 г.