Unión Internacional de Telecomunicaciones

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

J.1014

(04/2020)

SERIE J: REDES DE CABLE Y TRANSMISIÓN DE PROGRAMAS RADIOFÓNICOS Y TELEVISIVOS, Y DE OTRAS SEÑALES MULTIMEDIA

Acceso condicional y protección – Soluciones de acceso condicional insertadas e intercambiables y de gestión digital de los derechos

Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Seguridad avanzada – Funcionalidades específicas ECI

Recomendación UIT-T J.1014

Recomendación UIT-T J.1014

Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Seguridad avanzada – Funcionalidades específicas ECI

Resumen

La presente Recomendación forma parte de un conjunto de publicaciones que abarca las funcionalidades específicas ECI de un **sistema de seguridad avanzada** para la **interfaz común integrada (ECI)** a los efectos de la especificación de soluciones intercambiables de acceso condicional/gestión de derechos digitales (CA/DRM).

Esta Recomendación del UIT-T es una transposición de la norma [b-ETSI GS ECI 001-5-1], y es el resultado de la colaboración entre la CE 9 del UIT-T y el ISG ECI del ETSI. Se han introducido modificaciones en las cláusulas 1, 3.2, 6.1, 6.2, 6.3 y 8.2.3 y algunas correcciones editoriales, así como la sustitución del término "trayecto de vídeo seguro" por "trayecto de vídeo seguro".

Historia

Edición	Recomendación	Aprobación	Comisión de Estudio	ID único*
1.0	UIT-T J.1014	23-04-2020	9	11.1002/1000/13575

Palabras clave

CA, DRM, intercambio.

* Para acceder a la Recomendación, sírvase digitar el URL http://handle.itu.int/ en el campo de dirección del navegador, seguido por el identificador único de la Recomendación. Por ejemplo, http://handle.itu.int/11.1002/1000/11830-en.

PREFACIO

La Unión Internacional de Telecomunicaciones (UIT) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones y de las tecnologías de la información y la comunicación. El Sector de Normalización de las Telecomunicaciones de la UIT (UIT-T) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la AMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

La observancia de esta Recomendación es voluntaria. Ahora bien, la Recomendación puede contener ciertas disposiciones obligatorias (para asegurar, por ejemplo, la aplicabilidad o la interoperabilidad), por lo que la observancia se consigue con el cumplimiento exacto y puntual de todas las disposiciones obligatorias. La obligatoriedad de un elemento preceptivo o requisito se expresa mediante las frases "tener que, haber de, hay que + infinitivo" o el verbo principal en tiempo futuro simple de mandato, en modo afirmativo o negativo. El hecho de que se utilice esta formulación no entraña que la observancia se imponga a ninguna de las partes.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB en la dirección http://www.itu.int/ITU-T/ipr/.

ÍNDICE

Introducción

La presente Recomendación del UIT-T[1] es una transposición de la norma [b-ETSI GS ECI 001-5-1], y es el resultado de la colaboración entre la CE 9 del UIT-T y el ISG ECI del ETSI. Se han introducido modificaciones en las cláusulas 1, 3.2, 6.1, 6.2, 6.3 y 8.2.3 y algunas correcciones editoriales, así como la sustitución del término "trayecto de vídeo seguro" por "**trayecto de vídeo seguro**"[2].

El objetivo de esta Recomendación es facilitar la interoperabilidad y la competencia en los servicios de comunicaciones electrónicas y, en particular, en el mercado de los dispositivos de radiodifusión y audiovisuales. Sin embargo, existen otras tecnologías disponibles que también pueden resultar adecuadas y beneficiosas, en función de las circunstancias de los Estados Miembros.

La protección de servicios y contenidos mediante acceso condicional (CA) y gestión de derechos digitales (DRM) es esencial en el ámbito en continua evolución de la radiodifusión digital y la banda ancha, que abarca contenidos, servicios, redes y equipos en las instalaciones del cliente (CPE), con el fin de proteger los modelos de negocio de los propietarios de contenidos, los operadores de red y los operadores de TV de pago. También es fundamental que los consumidores puedan seguir utilizando los CPE previamente adquiridos, por ejemplo, tras un traslado o un cambio de proveedor de red, o incluso utilizar dispositivos para servicios de distintos portales de vídeo comerciales. Esto sólo es posible mediante mecanismos de CA y DRM interoperables incorporados en los CPE, sobre la base de una arquitectura de seguridad adecuada.

La presente Recomendación define como parte de la arquitectura de seguridad un sistema de procesamiento de seguridad para la autenticación y verificación del contenido de medios protegidos y de imágenes software que deben procesar los CPE conformes con ECI. El componente fundamental de la arquitectura de seguridad es el **Bloque de Escalera de Claves**, que soporta un procesamiento seguro con claves secretas, dedicando determinadas claves a circuitos integrados (chip) específicos y la autenticación del origen de recursos básicos.

La cláusula 6 presenta una visión general de la arquitectura del sistema, define reglas sobre la **Robustez** contra los ataques y describe la relación entre los elementos de la arquitectura de seguridad, el **Anfitrión ECI** y los **Clientes ECI**.

En la cláusula 7 se describen las aplicaciones para las que puede utilizarse el **Bloque de Escalera de Claves**, junto con las funciones asociadas.

Para un funcionamiento adecuado, el sistema de procesamiento de seguridad necesita información sobre el estado de cada **Cliente ECI** cargado. Esta información de estado, parte de la cual debe ser secreta, se maneja con ayuda de un intervalo de seguridad avanzada. El **Anfitrión ECI** asigna a cada **Cliente ECI** un intervalo, que debe estar protegido contra modificaciones maliciosas. La definición del intervalo y su configuración para diversas operaciones como la desencriptación o la exportación de contenido se describen en la cláusula 8.

En un CPE conforme con **ECI** se puede desencriptar el contenido, enviarlo a salidas normalizadas, si está permitido, y reencriptarlo para su exportación. En la cláusula 9 se describe la utilización de un intervalo de seguridad avanzada para esas operaciones.

Un **Subsistema de Procesamiento de Certificados** implementado como función especial de un intervalo de seguridad avanzada, es responsable de la autenticación de elementos. En la cláusula 10 se especifican las reglas de autenticación.

---

[1] En el Apéndice II se indican varios aspectos susceptibles de mejora.

[2] La letra negrita en el texto de esta Recomendación indica términos con definiciones específicas para el contexto de la interfaz común integrada que pueden diferir del uso común.

El sistema **ECI** utiliza un mecanismo cargador que permite a los **Clientes ECI** verificar de forma segura la versión de las credenciales cargadas del **Anfitrión ECI** y del **Cliente ECI**, de forma que pueda detectarse cualquier problema de seguridad reconocido. El mecanismo del cargador se basa en principios relativos a la **Robustez** descritos en la cláusula 11.

La cláusula 12 aborda las limitaciones en materia de temporización de las operaciones descritas en la presente Recomendación.

Recomendación UIT-T J.1014

Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Seguridad avanzada – Funcionalidades específicas ECI

1 Alcance

La presente Recomendación define un subsistema de procesamiento de seguridad robusto para **ECI** denominado **Sistema de Seguridad Avanzada**. El **Sistema de Seguridad Avanzada** proporciona una base segura para la autenticación y la carga de elementos software, realiza cálculos y verificaciones de seguridad, gestiona la encriptación y la desencriptación de contenido y el intercambio de contenidos con derechos y obligaciones asociados. El **Sistema de Seguridad Avanzada** utiliza el **Bloque de Escalera de Claves ECI** [UIT-T J.1015]. Para realizar cálculos seguros véase también [b-ETSI GS ECI 001-5-2].

2 Referencias

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes. En esta Recomendación, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación.

[UIT-T J.1010] Recomendación UIT-T J.1010 (2016), *Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Casos y requisitos de utilización*.

[UIT-T J.1011] Recomendación UIT-T J.1011 (2016), *Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Arquitectura, definiciones y visión general*.

[UIT-T J.1012] Recomendación UIT-T J.1012 (2020), *Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Contenedor, cargador, interfaces y revocación de CA/DRM*.

[UIT-T J.1013] Recomendación UIT-T J.1013 (2020), *Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Máquina virtual*.

[UIT-T J.1015] Recomendación UIT-T J.1015 (2020), *Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Sistema de seguridad avanzado – Bloque de Escalera de Claves*.

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

[ISO/CEI 23001-7] ISO/CEI 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/CEI 23009-4] ISO/CEI 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 Definiciones

3.1 Términos definidos en otros documentos

Ninguno.

3.2 Términos definidos en esta Recomendación

En esta Recomendación se definen los siguientes términos:

3.2.1 sistema de seguridad avanzada (Sistema AS): función de un **CPE** que es conforme con **ECI** que ofrece funciones de seguridad mejoradas (hardware y software) para un **Cliente ECI**.

3.2.2 intervalo de seguridad avanzada (Intervalo AS): recursos del bloque de seguridad avanzada que un **Anfitrión ECI** proporciona exclusivamente a un **Cliente ECI**.

3.2.3 API de AS: interfaz de programa de aplicación entre el **Cliente ECI** y su **Anfitrión ECI** que permite al **Cliente ECI** intercambiar información con su **Intervalo AS** y realizar operaciones en el mismo.

3.2.4 mecanismo de autenticación: función del **Bloque de Escalera de Claves** definida en [UIT-T J.1015] que permite al **Intervalo AS** proporcionar aplicaciones de clave segura para fines distintos a la desencriptación y encriptación de contenido, como por ejemplo, la autenticación.

3.2.5 hermano: otro **Hijo** del mismo **Padre**.

NOTA – **Padre**, **Hijos**, **Hermano**, hacen referencia a **Entidades** que gestionan **Certificados**.

3.2.6 certificado: estructura de datos definida en la cláusula 5 de [UIT-T J.1012], véase también [b-ETSI GS ECI 001-3], con una firma digital segura suplementaria que identifica una **Entidad**.

NOTA – El titular de la clave secreta de la firma atestigua la corrección de los datos (los autentica) firmándolos con su clave secreta. Su clave pública puede utilizarse para verificar los datos.

3.2.7 cadena de certificados: lista de **Certificados** que se autentican unos a otros incluida una **Lista de Revocación Raíz**.

NOTA – Normalmente, los **Certificados ECI** están acompañados de una **Lista de Revocación** que excluye los **Certificados** que no pueden ser validados.

3.2.8 subsistema de procesamiento de certificados (CPS): subsistema del **Anfitrión ECI** que verifica el **Certificado** y proporciona **Robustez** adicional contra la manipulación indebida.

3.2.9 hijo, hijos: **Entidad** (**Entidades**) a las que hace referencia un **Certificado** firmado por un **Padre** (común).

NOTA – **Padre, Hijos, Hermano,** hacen referencia a entidades que gestionan **Certificados**: datos y software de inicialización utilizados para arrancar el SoC de un CPE.

3.2.10 propiedades del contenido (CP): propiedades del contenido que proporcionan información sobre derechos y obligaciones asociadas a subsiguientes aplicaciones o transformaciones del contenido, tal como información sobre derechos de uso, control de salida selectiva e información de control parental.

3.2.11 ECI (interfaz común integrada): arquitectura y sistema especificado por el ISG de ETSI "Embedded CI" (CI integrada), que permite la creación e instalación en equipos de cliente (**CPE**) de **Clientes ECI** intercambiables basados en software y que, por lo tanto, permiten la interoperabilidad de dispositivos **CPE** con relación a la **ECI**.

3.2.12 cliente ECI (cliente de la interfaz común integrada): implementación de un cliente de CA/DRM conforme con las especificaciones **ECI**.

3.2.13 cargador de cliente ECI: funcionalidad del **Anfitrión ECI** que utiliza el **Sistema AS** para proporcionar de forma exclusiva la función, la verificación y la instalación de un nuevo software de **Cliente ECI** en un contenedor **ECI** del **Anfitrión ECI**.

3.2.14 ecosistema ECI: operativa comercial que consta de una **TA** (autoridad confiable) y varias plataformas y **CPE** conformes con **ECI**.

3.2.15 anfitrión ECI: sistema hardware y software de un **CPE**, que abarca funcionalidades relativas a la **ECI** y que tiene interfaces con un **Cliente ECI**.

NOTA – El **Anfitrión ECI** forma parte del firmware del CPE.

3.2.16 cargador de anfitrión ECI: funcionalidad para la descarga y el arranque del CPE que utiliza el **Sistema AS** para proveer de forma exclusiva la función de verificación e instalación del software del **Anfitrión ECI** en un CPE.

NOTA – En una configuración de descarga en varias fases este término se utiliza para hacer referencia a todas las funciones de la descarga críticas en cuanto a la seguridad que participan en la carga del **Anfitrión ECI**.

3.2.17 clave raíz ECI: clave pública que proporciona el origen de autenticación para entidades certificadas **ECI** y **Certificados**.

3.2.18 entidad: organización (por ejemplo, fabricante, **Operador** o **Suministrador de seguridad**) o elemento del mundo real (por ejemplo, **Anfitrión ECI**, **Operación de Plataforma** o **Cliente ECI**) identificado mediante un ID único en un **Ecosistema ECI**.

3.2.19 conexión de exportación: relación autenticada entre un **Intervalo AS** que desencripta el contenido y otro **Intervalo AS** que subsiguientemente reencripta el contenido desencriptado indicando que dicha reencriptación está permitida.

3.2.20 padre: firmante del **Certificado** de la **Entidad hijo**.

NOTA – **Padre, Hijos, Hermano**, hacen referencia a entidades que gestionan **Certificados**.

3.2.21 escalera de claves: función del **Bloque de Escalera de Claves** definido en [UIT-T J.1015] para calcular las palabras de control y la información de uso de las palabras de control asociadas para su aplicación en la función de desencriptación o de reencriptación del contenido de un CPE.

3.2.22 bloque de escalera de claves: mecanismo seguro y robusto para calcular las claves de desencriptación, encriptación y autenticación definidas en [UIT-T J.1015], tanto de la **Escalera de Claves** como del **Mecanismo de Autenticación**.

3.2.23 microcliente: **Cliente ECI** o no **ECI** que puede desencriptar contenido que ha sido reencriptado por un **Microservidor**.

3.2.24 sistema microDRM: sistema de protección del contenido que reencripta contenido en un CPE con un **Microservidor** y que permite la decodificación de ese contenido reencriptado por **Microclientes** autenticados.

NOTA – El Microservidor y el Microcliente son aprovisionados por un Operador de sistema microDRM.

3.2.25 microservidor: **Cliente ECI** que puede importar contenido desencriptado, encriptarlo de nuevo y autenticar un **Cliente ECI** específico o un grupo de **Clientes ECI** como **Objetivo** para una ulterior desencriptación.

3.2.26 operador: organización que proporciona **Operaciones de Plataforma** que la **TA ECI** reconoce para la firma del **Ecosistema ECI**.

NOTA – Un **Operador** puede explotar varias **Operaciones de Plataforma**.

3.2.27 operación de plataforma (PO): instancia específica de una operación de prestación de un servicio técnico con una única identidad **ECI** relacionada con la seguridad.

3.2.28 servidor de aprovisionamiento: servidor, normalmente ubicado en una instalación de apoyo que proporciona claves y otra información segura para facilitar una función de encriptación o desencriptación en un **Intervalo AS**.

3.2.29 revocación: situación de exclusión de una **Entidad** al estar incluida en una **Lista de Revocación**.

3.2.30 lista de revocación (RL): lista de **Certificados** que han sido revocados y que por lo tanto no deberían seguir utilizándose.

3.2.31 robustez: propiedad de la implementación de una función **ECI** segura especificada que representa el esfuerzo y el costo para poner en riesgo la seguridad de la función segura implementada.

3.2.32 certificado raíz: certificado confiable que es el origen de la autenticación para una cadena de certificados en un **Ecosistema ECI**.

3.2.33 trayecto de vídeo seguro: todas las funciones del CPE que realizan el procesamiento del contenido (y el almacenamiento temporal necesario para ello) desde el desencriptado de contenido hasta el reencriptado de contenido, ambos inclusive, por medio de un microcliente o un sistema de protección de salida.

3.2.34 suministrador de seguridad: empresa que suministra sistemas de seguridad **ECI**, incluyendo **Clientes ECI**, para **Operadores** de **Operaciones de Plataforma**.

3.2.35 objetivo: **Microcliente** o grupo de **Microclientes** para los que un **Microservidor** reencripta el contenido.

4 Abreviaturas y acrónimos

En esta Recomendación se utilizan las abreviaturas y acrónimos siguientes:

ACF Campo de control de seguridad avanzada (*advanced security control field*)

AD Datos relacionados (*associated data*)

AES Norma de encriptación avanzada (*advanced encryption standard*)

AK Clave de autenticación (*authentication key*)

API Interfaz de programación de aplicaciones (*application programming interface*)

ARK Clave aleatoria de seguridad avanzada (*advanced security random key*)

AS Seguridad avanzada (*advanced security*)

CA	Acceso condicional (*conditional access*)
CBC	Cadena de bloque de cifrado (*cypher block chaining*)
CENC	Encriptación común (*common encryption*)
CISSA	Algoritmo de aleatorización orientado al software común de IPTV (*common IPTV software-oriented scrambling algorithm*)
CP	Propiedades del contenido (*content property*)
CPE	Equipo en los locales del cliente (*customer premises equipment*)
CPS	Subsistema de procesamiento de certificados (*certificate processing subsystem*)
CSA	Algoritmo de aleatorización común (*common scrambling algorithm*)
CTR	Modo contador (*counter mode*)
CW	Palabra de control (*control word*)
DRM	Gestión de derechos digitales (*digital rights management*)
EAC	Certificado de autorización de exportación (*export authorization certificate*)
EAOC	**Certificado de operador** de autorización de exportación (*export authorization **operator certificate***)
ECI	Interfaz común integrada (*embedded common interface*)
ECM	Mensaje de control de prerrogativas (*entitlement control message*)
ECP	Protección de contenido mejorada (*Enhanced Content Protection*)
EGC	Certificado de grupo de exportación (*export group certificate*)
ERC	Certificado de **revocación** de exportación (*export **revocation** certificate*)
ESC	Certificado de sistema de exportación (*export system certificate*)
LK	Clave de enlace (*link key*)
MPEG	Grupo de expertos en imágenes en movimiento (*motion picture experts group)*
MSCSK	Clave secreta de circuito integrado del microservidor (*micro server chipset secret key*)
PES	Flujo elemental en modo paquete (*packetized elementary stream*)
PO	**Operación** de plataforma (*platform **operation***)
POC	Certificado de **operación de plataforma** (***platform operation** certificate*)
POPK	Clave pública de **operación de plataforma** (***platform operation** public key*)
REAOC	Certificado de **revocación de operador** de autenticación de exportación (***revocation** export authentication **operator** certificate*)
REE	Entorno de ejecución enriquecido (*rich execution environment*)
RFU	Reservado para uso futuro (*reserved for future use*)
RK	Clave aleatoria (*random key*)
RL	**Lista de revocación** (***revocation list***)
SPK	Clave pública del emisor (*sender public key*)
TA	Autoridad de confianza (*trust authority*)
TEE	Entorno de ejecución fiable (*trusted execution environment*)
TLS	Seguridad en la capa de transporte (*transport layer security*)

TPEGC Certificado de grupo de exportación de tercero (*third party export group certificate*)

TS Flujo de transporte MPEG-2 (*MPEG-2 transport stream)*

URI Información de derechos de uso (*usage rights information)*

XT Campo eXTensión (*eXTension field*)

5 Convenios

La utilización en la presente Recomendación de términos escritos en negrita y con la primera letra en mayúsculas refleja que dichos términos tienen un significado específico a efectos de la ECI que puede diferir del uso habitual de los mismos.

6 Principios

6.1 Visión general

La presente Recomendación pertenece a una serie de Recomendaciones UIT-T basadas en la arquitectura **ECI** [UIT-T J.1011], véase también [b-ETSI GS ECI 001-1], y en los requisitos básicos **ECI** [UIT-T J.1010], véase asimismo [b-ETSI GS ECI 001-2].

En la Figura 6-1 se presentan los principios básicos del **Sistema de Seguridad Avanzada**. El núcleo del **Sistema de Seguridad Avanzada** está formado por el **Bloque de Escalera de Claves** tal como se define en [UIT-T J.1015], que permite un procesamiento seguro con claves secretas, destinado a disponer de claves para circuitos integrados específicos y la autenticación del origen de recursos fundamentales.

La base para la carga de imágenes reside en el **núcleo del cargador**. Este utiliza el **Subsistema de Procesamiento de Certificados** para verificar el estado **ECI** de las imágenes de **Anfitrión ECI**, las imágenes de **Cliente ECI** y las credenciales de **Operación de Plataforma** (PO) utilizando una **Clave Raíz ECI** reciente y una **Lista de Revocación** raíz **ECI**. Los números de versión de la **Clave Raíz ECI** y de la **Lista de Revocación** raíz **ECI** utilizadas por el **Anfitrión ECI** y por otros **Clientes ECI** puede ser verificada por **Clientes ECI** cargados. Estos pueden rechazar la desaleatorización de contenido al detectar versiones inaceptables de conformidad con el principio de aplicación de la **Revocación ECI**. Las imágenes de **Cliente ECI** encriptadas se desencriptan al ser cargadas.

Todos los **Clientes ECI** utilizan un intervalo de seguridad avanzada. El **Intervalo AS** se identifica mediante la Clave Pública de la **Operación de Plataforma** del **Cliente ECI**. El **Anfitrión ECI** garantiza que las interacciones de un **Cliente ECI** mediante la **API de AS** (API de seguridad avanzada) se dirigen al **Intervalo AS** asignado a ese **Cliente ECI**. Cada **Intervalo AS** se describe mediante un estado del intervalo y un estado de la sesión para cada operación de encriptación/desencriptación. El **Intervalo AS** puede utilizarse con fines de desencriptación o de encriptación. El estado de la sesión del **Intervalo AS** también incluye una configuración (config) que define aspectos detallados de la operación y cómo debería autenticarse el estado de la sesión. El **Cliente ECI** proporciona la información de configuración y de entrada para otra información de estado. El **Bloque de Escalera de Claves** se utiliza para autenticar la clave pública del emisor (SPK), la clave pública de operación de plataforma (POPK) y la información de configuración. El **Intervalo AS** puede suministrar números aleatorios a entradas seleccionadas del **Bloque de Escalera de Claves** a fin de generar claves aleatorias o usarlas como palabras de ocasión (palabra aleatoria de un solo uso, que en el idioma inglés se denomina "nonce") que garanticen que las entradas del **Bloque de Escalera de Claves** han sido calculadas recientemente. Este mecanismo puede utilizarse para impedir la reproducción de contenidos encriptados y para garantizar el suministro en línea de un **Intervalo AS** por un **Servidor de Aprovisionamiento**.

Cuando se desencripta el contenido, las **Propiedades del Contenido** pueden autenticarse al tiempo que se calculan las palabras de control, creando así un fuerte vínculo con el contenido desencriptado. Las **Propiedades del Contenido** se envían junto con el contenido a cualquier salida normalizada para garantizar valores adecuados de los mecanismos de protección para dicha salida. Estas propiedades se comparan con aquellas con las que se reencripta el contenido en un **Conexión de Exportación**. Sólo es posible establecer una **Conexión de Exportación** a través de las **Cadenas de Certificados** de exportación/importación adecuadas. Estas son verificadas por el **Subsistema de Procesamiento de Certificados** en nombre del **Intervalo AS**. Las salidas normalizadas pueden inhabilitarse mediante el mecanismo de control de salida.

Las palabras de control calculadas pueden sincronizarse con el contenido con formato de flujo de transporte MPEG mediante el protocolo de alternancia de bits. Por este motivo, el **Trayecto de Vídeo Seguro** utiliza un mecanismo de doble almacenamiento en memoria con una palabra de control actual/siguiente para el procesamiento del flujo.

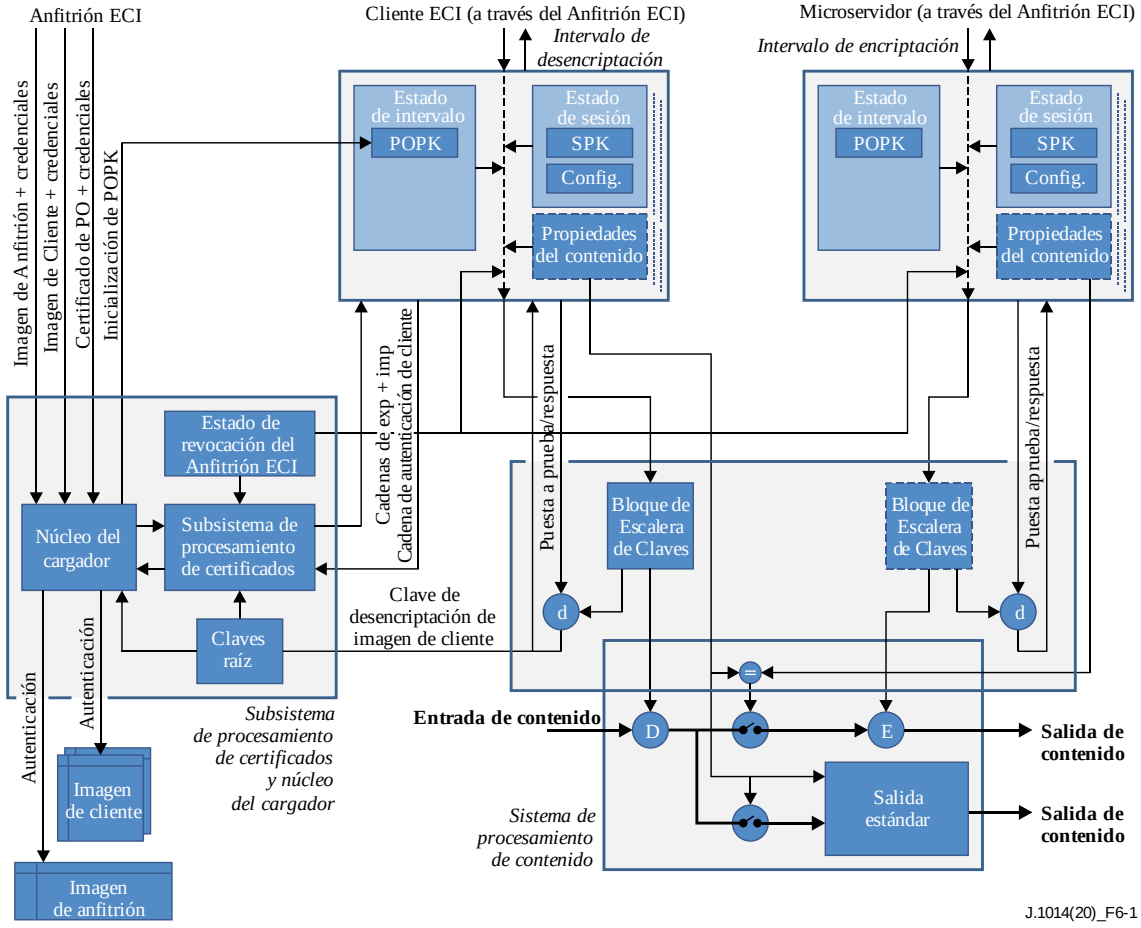

Figura 6-1 – Visión esquemática del Sistema de Seguridad Avanzada

6.2 Modelo de la robustez del sistema

El **Sistema AS** requiere una implementación robusta. La **Robustez** se mide normalmente en términos del esfuerzo y/o el coste requerido para burlar alguna de las medidas de seguridad; es decir, observar valores secretos o manipular el estado o los valores de un sistema seguro.

La presente Recomendación no define un régimen de **Robustez** específico para diversas funciones **ECI**. No obstante, la arquitectura de **Robustez ECI** se basa en la premisa de que algunas funciones son más robustas que otras. Esto se ilustra en la Figura 6-2.

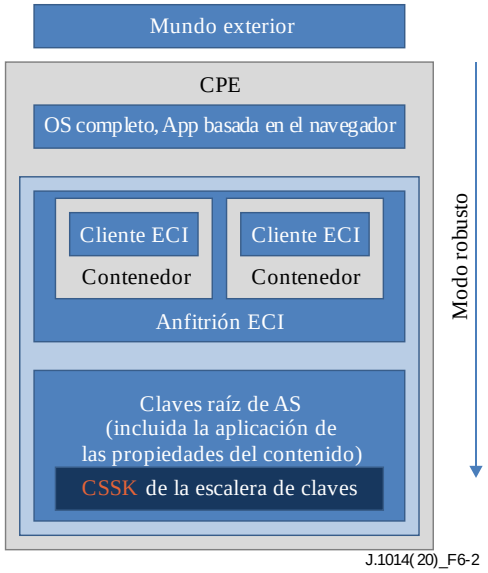

Figura 6-2 – Premisas de la Robustez del sistema para la ECI

El entorno menos robusto es el mundo exterior, donde puede existir cualquier tipo de amenaza. El paso de los datos por ese entorno debería estar protegido contra la manipulación e inspección no autorizada mediante técnicas de autenticación y encriptación. El sistema operativo en su totalidad (normalmente incluyendo un navegador) puede ser robusto hasta cierto punto contra la manipulación e invasión, pero normalmente no tiene capacidad para soportar ataques de usuarios o ataques externos agresivos de piratería informática. Las funciones sensibles a la seguridad de los **Clientes ECI** y el **Anfitrión ECI** operan en un entorno adecuadamente protegido de dichos ataques. Si un **Anfitrión ECI** se ve comprometido, también los estarán los **Clientes ECI**. Además de su resistencia contra ataques externos, los **Clientes ECI** están protegidos por el aislamiento que les ofrece la máquina virtual **ECI** [UIT-T J.1013], véase también [b-ETSI GS ECI 001-4]: es decir, no pueden acceder a información del **Anfitrión ECI** ni de otro **Cliente ECI**, distinta a la que permiten las interfaces de la API **ECI**. El **Anfitrión ECI** también garantiza que los **Clientes ECI** tengan acceso al **Sistema de Seguridad Avanzada** y al **Bloque de Escalera de Claves**. En la parte nuclear del **Bloque de Escalera de Claves** se encuentra la **Clave Secreta del Conjunto de Circuitos Integrados** que permite direccionar a cada **CPE ECI** de forma unívoca. Por lo general, el **Bloque de Escalera de Claves** y las partes principales del **Sistema de Seguridad Avanzada** se implementan en hardware o mediante un firmware muy robusto.

Si bien en la práctica la robustez de los componentes de seguridad es un valor continuo, desde el punto de vista arquitectónico pueden establecerse una jerarquía. En el contexto de la ECI, se puede describir del modo siguiente:

0) El mundo exterior (fuera del dispositivo) tiene un nivel de robustez igual a 0. Por lo tanto, el **Cliente ECI** necesita un protocolo seguro para comunicarse con la pasarela. Esto queda fuera del alcance de la presente Recomendación sobre **ECI**.

1) El sistema operativo, los controladores, las aplicaciones y el navegador del dispositivo tienen un nivel de robustez 1. El entorno en el que se ejecuta este código se suele denominar entorno de ejecución enriquecido (REE) y consta de la mayoría de los programas y controladores de las aplicaciones y del sistema operativo. Dado el inmenso tamaño y funcionalidad de este software, suele ser necesario parchearlo frecuentemente para eliminar vulnerabilidades. El **Anfitrión ECI** y el **Cliente ECI** pueden utilizar los servicios proporcionados por el REE, como la conexión en red TLS, pero no pueden confiar en él para cuestiones de seguridad, ya sea el encriptado o la autenticación de los puntos extremos.

2) El entorno de ejecución de funciones del **Anfitrión ECI**, la VM y el **Cliente ECI** deben ejecutarse en un entorno de ejecución seguro, a menudo en el mismo procesador. En el nivel de robustez 2, estos entornos sólo ejecutan código autenticado, tienen memoria forzada por hardware y aislamiento de dispositivos respecto del REE, incluyendo su núcleo y controladores. Estos se denominan a veces entornos de ejecución fiables (TEE). Este nivel también podría implementarse en un procesador específico o un procesador de seguridad con aislamiento de memoria.

3) El **Trayecto de Vídeo Seguro** precisa de un nivel de seguridad superior al del **Anfitrión ECI** y del **Cliente ECI**, es decir, el nivel de robustez 3. Puede implementarse combinando hardware y firmware seguros.

4) Los más seguros y robustos son los subsistemas de **Escalera de Claves**, el cargador de arranque y de procesamiento y revocación de certificados, que funcionan con un nivel de robustez 4. Lo ideal es que estén integrados en un hardware seguro, pero debido a la naturaleza de la Recomendación sobre **ECI**, algunas partes pueden requerir un firmware que se ejecuta en un procesador de seguridad específico. Se requieren medidas específicas para preservar el secreto de la seguridad avanzada y los secretos de la escalera de claves y los cálculos correspondientes.

6.3 Trayecto de Vídeo Seguro y Control del Sistema de Protección de Salida

Una vez desencriptado el contenido, el **Trayecto de Vídeo Seguro** lo protege contra el uso ilegítimo. El **Trayecto de Vídeo Seguro** puede realizar diversas operaciones de vídeo (y audio) sobre el contenido no encriptado: demultiplexación, descodificación, escalada, recodificación, inserción de filigranas, etc. Todas las operaciones y sus partes integrantes (incluidas las memorias transitorias implicadas y los buses utilizados para transferir el contenido entre diferentes subsistemas del CPE), así como las interfaces de control (propiedades del contenido y posibles interfaces específicas de la implementación o de propiedad) que controlan las distintas etapas esenciales del procesamiento de las propiedades del contenido y los ajustes de protección de salida del contenido se consideran parte del **Trayecto de Vídeo Seguro**.

Las propiedades del contenido [UIT-T J.1012] definen qué operaciones están permitidas y qué salidas, sistemas de protección de salida y sus ajustes son adecuados para proteger el contenido. El sistema de seguridad avanzada proporciona un mecanismo criptográfico denominado autenticación de las propiedades del contenido (véase la cláusula 8.2.3) para autenticar implícitamente las propiedades del contenido utilizando el bloque de escalera de claves al calcular la palabra de control. El sistema de protección de contenidos puede seleccionar la configuración de la autenticación de las propiedades del contenido. Las **Propiedades del Contenido** no autenticadas implícitamente por la función de autenticación de las propiedades del contenido tienen el nivel de robustez del **Cliente ECI** y del **Anfitrión ECI**.

Cabe señalar que se necesitan especificaciones complementarias para proporcionar detalles adecuados sobre la solidez de la implementación del trayecto de vídeo seguro y los sistemas de protección de salida.

La Figura 6-3 describe en detalle de la lógica para garantizar el suministro de las **Propiedades del Contenido**. En primer lugar el **Cliente ECI** establece las **Propiedades del Contenido** de la sesión para la siguiente sección de contenido que se va a decodificar. A continuación, el **Cliente ECI** introduce los datos para calcular la palabra de control, incluido el Field1 (como parte del vector de entrada **elk**: véanse las cláusulas 7.1, 8.2.3, 8.2.4.7). La función de autenticación de las propiedades del contenido selecciona los dos primeros bytes del Field1, donde se encuentran los campos de las propiedades del contenido del Field1, y posteriormente se autentican implícitamente utilizando este valor como parámetro de valor de clave simétrica para el cálculo de la escalera de claves de CW. Si el valor de clave simétrica es incorrecta el CW será erróneo, por lo que el contenido no podrá ser desencriptarse. El SVP compara los valores de Field1 con las propiedades de contenido

correspondientes establecidas por el Cliente ECI. En caso de discrepancia, no será posible obtener una salida. La salida del sistema de protección específico también puede quedar bloqueada por el correspondiente valor del campo de control de salida.

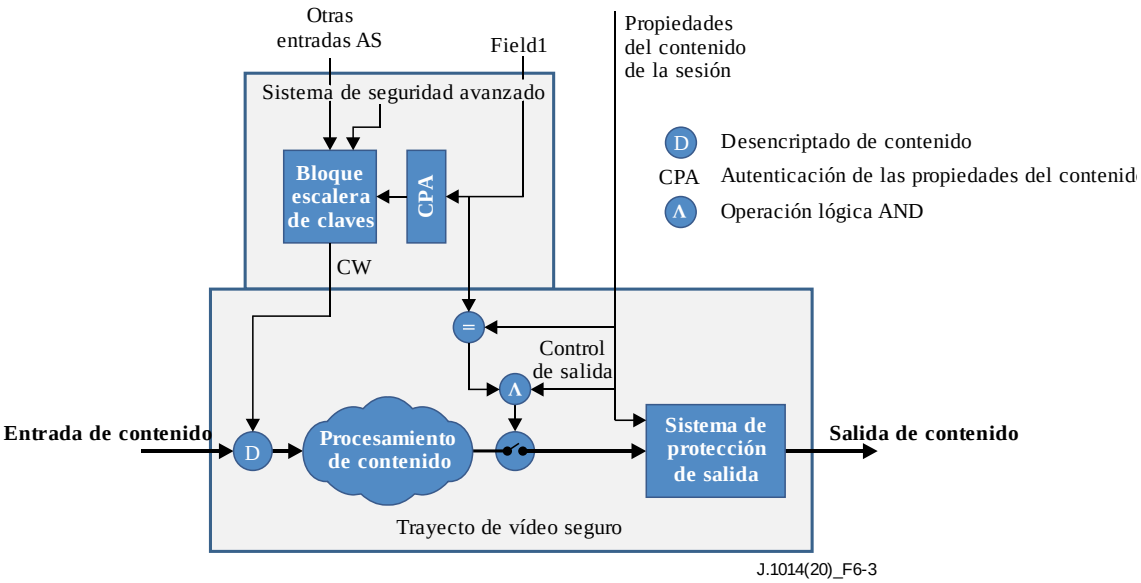

Figura 6-3 – Trayecto de video seguro y autentificación de la propiedad del contenido

6.4 Principios de la especificación

6.4.1 Libertad de implementación

En la presente Recomendación se definen estados y funciones que operan en el **Sistema AS,** dando lugar a un nuevo estado. No se define la representación específica del estado de una implementación, que puede quedar completamente definido por la propia implementación en la medida en que el comportamiento de la misma pueda describirse con estados y secuencias de transición de estado utilizando las funciones de transición definidas en esta Recomendación.

NOTA – En muchos casos, la función **Escalera de Claves**, tal como se define en [UIT-T J.1015], es una parte sustancial de la función de transición de estado.

Por ejemplo, una implementación AS puede tener un subsistema de procesamiento de certificados (CPS) rápida que pueda reautenticar la POPK de la **Cadena de Certificados de Operación de Plataforma** para cada aplicación del **Bloque de Escalera de Claves**. En este caso, el **Intervalo AS** no tiene que almacenar la POPK como un valor autenticado a prueba de manipulaciones. De la misma forma, algunas implementaciones pueden decidir calcular LK_1 (clave simétrica de nivel superior en la **Escalera de Claves**) una sola vez mediante dos operaciones criptográficas asimétricas para un **Intervalo AS**, mientras que otras que pueden realizar las operaciones criptográficas asimétricas con suficiente rapidez, pueden recalcular LK_1 a partir de sus entradas originales para cada aplicación de la **Escalera de Claves**.

6.4.2 Estilo de la especificación y relación con la API de AS

No existe una API directa entre el **Cliente ECI** y el **Sistema de Seguridad Avanzada**. El **Anfitrión ECI** actúa como conducto entre ellos. No obstante, las definiciones de las operaciones en un **Intervalo AS** se corresponden directamente con los mensajes de la **API AS** tal como se define en [UIT-T J.1012], con la excepción del parámetro slotId, que no es necesario para los mensajes API de AS del **Cliente ECI**. El **Anfitrión ECI** proporciona el parámetro slotId al **Sistema AS**.

Las transacciones del **Anfitrión ECI** (en nombre de un **Cliente ECI**) en un **Intervalo AS** se definen como declaración de funciones en notación C. Describen una transacción atómica relativa al estado del **Intervalo AS**. Ello puede dar lugar a un nuevo estado del intervalo. La representación específica de parámetros de función no es consecuencia directa de una funcionalidad especificada en la presente Recomendación, excepto cuando están involucradas funciones criptográficas. No obstante, la representación es importante para la definición de la **API de AS** en [UIT-T J.1012].

7 Aplicación de la Escalera de Claves y funciones asociadas

7.1 Generalidades

La **Escalera de Claves** y el **Mecanismo de Autenticación** definidos en [UIT-T J.1015] juegan un papel fundamental en todos los cálculos de clave secreta que se implementan de forma robusta en un **CPE ECI**. El **Sistema de Seguridad Avanzada** aplicará esas funciones tal como se define en [UIT-T J.1015], con entradas y salidas descritas en la presente Recomendación. Todas las entradas al **Bloque de Escalera de Claves** estarán controladas por el **Sistema AS**; no será posible ninguna observación o manipulación con arreglo a las reglas de **Robustez** AS y a la especificación del **Bloque de Escalera de Claves** [UIT-T J.1015].

7.2 Sistema AS y autenticación de datos de cliente

Un **Cliente ECI** puede proporcionar datos a un **Sistema de Seguridad Avanzada**. El **Sistema AS** ofrece una forma de verificar la autenticidad de estos datos utilizando la entrada AD del **Bloque de Escalera de Claves**.

El **Sistema AS** calcula la entrada AD del **Bloque de Escalera de Claves** como la función hash de los datos adicionales que deben ser autenticados conjuntamente con un cálculo de una CW o una AK. Conforme a la notación de cadena de bits [UIT-T J.1015], la AD se calculará de la forma siguiente:

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

La función *hash* (troceo y aleatorización) se define en la cláusula A.1. El factor lm es una entrada de 8 bits que contiene la representación binaria de m. El valor de m corresponde al valor de m en la definición del **Bloque de Escalera de Claves**. La longitud de cada P_i es de 2 048 bits para el transporte de las claves públicas (valores de POPK). La longitud de C_i se define como la longitud de la estructura SessionConfig de la cláusula 8.2.2.5, la longitud de XT es 256 y sirve a un mecanismo de extensión de propósito general. Se pondrá a 0. ARK es un número de 128 bits, que representa un valor aleatorio o bien, todos cero en caso de que no sea necesaria una entrada aleatoria. ACF es un valor de control que define el modo de funcionamiento.

7.3 Modo Microservidor asimétrico

El **Sistema AS** aplica el **Mecanismo de Autenticación** para cargar una clave secreta de emisor del **Microservidor** en un **Intervalo AS** al objeto de realizar una autenticación asimétrica entre **Microservidores** y **Microclientes**.

En la Figura 7-1 se muestra el principio básico de cálculo general del modo de autenticación asimétrica.

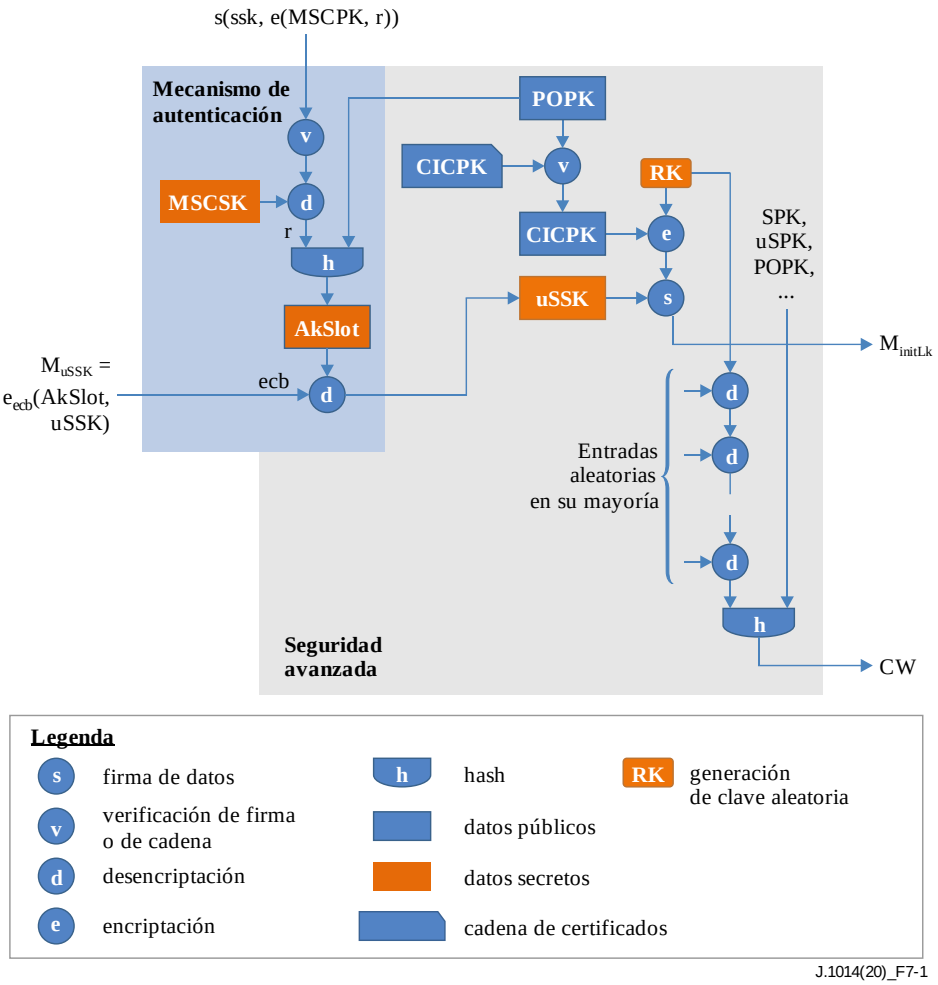

Figura 7-1 – Cálculo del modo Microservidor asimétrico

El **Mecanismo de Autenticación** se utiliza para calcular la clave de autenticación AkSlot del **Intervalo AS**, utilizando, entre otros, la clave del conjunto de circuitos integrados que en este caso se denomina Clave secreta de circuito integrado del microservidor (MSCSK). AkSlot se utiliza para cargar la clave secreta del Microservidor, uSSK. El **Subsistema de Procesamiento de Certificados** se utiliza para autenticar la clave pública del conjunto de circuitos integrados (CICPK) del **Microcliente Objetivo** utilizando la POPK como raíz y una **Cadena de Certificados** que contenga la CICPK en el último **Certificado** de la cadena. Se genera una clave aleatoria RK y CICPK y uSSK se utilizan para generar el mensaje de inicialización de la **Escalera de Claves del Microcliente,** M_{initLk}. La clave aleatoria también se utiliza como la clave simétrica de nivel superior de una **Escalera de Claves** con la estructura y la función *hash* definida en la cláusula 7.1 de [UIT-T J.1015]. La palabra de control (CW) calculada se utiliza para la encriptación que realiza el **Microservidor**. La **Escalera de Claves** ordinaria puede utilizarse en un **Microcliente** al objeto de calcular la CW con fines de desencriptación.

En la cláusula 8.2.4.10 se definen las características específicas del cálculo de uSSK (la función **ldUssk**).

El cálculo de M_{initLk} debe ser como se define en el esquema de cálculo siguiente, aplicando los convenios de especificación de la cláusula 7.1 de [UIT-T J.1015]:

- Mkey = cl-chipset-ID || E(ClCPK,LK)
- M_{initLk} = (Mkey || S(uSSK,Mkey))

- siendo || la función de concatenación a nivel de bits, cl-chipset-ID el ID del chipset (conjunto de circuitos integrados) del CPE cliente, E() la función de encriptación asimétrica y S() la función de firma asimétrica tal como se define en la cláusula 7.2 de [UIT-T J.1015].

Al cargar la uSSK. El **Intervalo AS** generará una nueva RK con arreglo a la cláusula A.3.

El cálculo de la CW a partir de LK y sus entradas será idéntico al Mecanismo de **Escalera de Claves** definido en la cláusula 7.2 de [UIT-T J.1015], con la mismas entradas que allí se definen y la misma salida (CW, CW-URI), pero sustituyendo el cálculo de LK1 por el de la clave de sesión aleatoria RK generada por el **Intervalo AS** del **Microservidor**.

7.4 Interfaz con el trayecto de vídeo seguro

La **Escalera de Claves**, incluyendo la extensión del modo **Microservidor** asimétrico, puede calcular las palabras de control con CW-URI suplementaria. Estas pasarán de forma segura a un recurso de desencriptación o encriptación en el subtrayecto de vídeo seguro que puede almacenar (temporalmente) la información de CW y CW-URI asociada a la información de sincronización de clave. Para aplicaciones de flujo de transporte, la información de sincronización de clave consta del bit actual/siguiente, especificando, por lo tanto, dos ubicaciones de almacenamiento para valores de CW. Para aplicaciones basadas en ficheros, sólo hay disponible una ubicación de CW para el recurso de encriptación y desencriptación.

Otra información que acompaña a las palabras de control desde el **Intervalo AS** al **Trayecto de Vídeo Seguro** es la siguiente:

- Aplicación de una CW como palabra de control de encriptación o desencriptación.

NOTA – Junto con la CW-URI constituye un permiso para aplicar la CW.

- Información de autenticación de exportación.
- Propiedades del contenido.
- Propiedad impar/par de la palabra de control para desaleatorización en modo TS.

7.5 Definición de entradas y salidas del Bloque de Escalera de Claves de AS

En esta cláusula se define la correspondencia de variables y estructuras en notación C como representación de las entradas a la **Escalera de Claves** y a los **Mecanismos de Autenticación** y las extensiones de los mismos tal como se define en las cláusulas 7.2 y 7.3. Los símbolos de los nombres de las entradas se utilizan en el resto de esta Recomendación para definir las diversas aplicaciones.

La correspondencia entre estructuras en notación C y una secuencia de octetos se define en virtud de las reglas siguientes (basadas en el esquema de ordenación en memoria de la información denominada "little endian", en la que los bits menos significativos se almacenan en primer lugar):

- Los campos de bits de las estructuras de datos se organizan colocando en primer lugar el bit menos significativo (0) del primer octeto.
- Las estructuras de datos de una longitud que no sea múltiplo de 8 bits se rellenan con ceros hasta que el número de bits alcance el siguiente múltiplo de 8.
- Las entidades de 16, 32 y 64 bits se organizan en orden "little endian" (el byte menos significativo en primer lugar).
- Las matrices se organizan en orden de índice creciente.

Las secuencias de octetos se organizan en cadenas de bits utilizando la función OS2BSP definida en [UIT-T J.1015].

NOTA – Las reglas anteriores garantizan que el orden de numeración de bits utilizado para valores enteros representados mediante variables en notación C es igual al utilizado en [UIT-T J.1015] para las correspondientes entradas al **Bloque de Escalera de Claves**.

El convenio de denominación de las variables se muestra en el Cuadro 7-1.

Cuadro 7-1 – Convenio de denominación de variables en notación C para la interfaz de la Escalera de Claves

Entrada o salida del Bloque de Escalera de Claves	Bits	Convenio de denominación de variables en notación C
CW-URI	64	ulong **cwUri**;
AD	256	uchar **ad**[32]; /* no utilizado directamente, véase la cláusula 7.2 */
SPK-URI	64	ulong **spkUri**;
SPK_i, i=1..16	2 048 × 16	typdef uchar PubKey[256]; PubKey **spk**[16]; /* (spk[i-1] == SPK_i) */
m		uchar **nSpk**;
input to V	64+ 2 048 + 2 048	typedef struct InputV{ ulong chipsetId; uchar elk1[256]; uchar signature[256]; } InputV; InputV;
$E(LK_i,LK_{i+1})$, i=1..24; LK_{t+1}=r	256 × 24	typedef uchar SymKey[32]; Symkey **elk**[i]; /* (elk[i-1] == $E(LK_i,LK_{i+1})$) */ /* **C-input** == $E(LK_{t-1},LK_{t1})$, es decir, la penúltima entrada ***/**
t		uchar **nElk**;
τ_b		value set to 0
Chipset-ID	64	ulong **chipsetId**;
Challenge	128	uchar **challenge**[16];
Response	128	uchar **response**[16];
A continuación se recogen las entradas y salidas definidas en la presente Recomendación.		
ACF	128-8	uchar **acf**[15]; /* modo de operación */
ARK	128	uchar **ark**[16];
P_i	2 048 × 32	PubKey **pk**[32]; /* primer se aplican los valores de **m** */
C_i	sizeof(SessionConfig) × 32	SessionConfig **config**[?]; /* SessionConfig se define en la cláusula 8.2.2.6 */
XT	256	uchar **XT**[32]; /* el valor siempre está puesto a 0 */
M_{initLk}	64+ 2 048 + 2 048	InputV **mInitLk**;

Las siguientes funciones en notación C se definen utilizando las variables de entrada anteriores para obtener resultados.

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

Semántica:

Esta función calcula la función del bloque V y del bloque C en la **Escalera de Claves** para obtener lk1.

En el caso de servidor asimétrico, la función siguiente calcula el mensaje de inicialización para el **Microcliente Objetivo,** tal como se define en la cláusula 7.3:

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

Semántica:

Esta función calcula el mensaje de inicialización initLk1 con arreglo a la cláusula 7.3.

Las restantes funciones de la **Escalera de Claves** se ejecutan mediante:

```
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])
```

Semántica:

Esta función calcula el resto de la **Escalera de Claves** utilizando lk1 como resultado del bloque D en la **Escalera de Claves** a fin de producir una CW para el **Trayecto de Vídeo Seguro**.

Las funciones del **Mecanismo de Autenticación** para calcular una clave AK se realizan mediante:

```
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])
```

Semántica:

Esta función calcula el **Mecanismo de Autenticación** hasta AK y entrega el resultado.

A fin de utilizar la clave AK calculada, se define la siguiente función utilizando la interfaz puesta prueba-contestación y el bloque funcional d en el **Mecanismo de Autenticación** de [UIT-T J.1015], cláusula 8:

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

Semántica:

Esta función calcula la contestación a una puesta a prueba de entrada utilizando AK como clave de autenticación, tal como se define en el **Mecanismo de Autenticación**.

7.6 Definición de ACF

La entrada campo de control de seguridad avanzada (ACF) al **Bloque de Escalera de Claves** permite definir los modos principales del modo de operación. El valor del parámetro acf[0] se define en el Cuadro 7-2.

Cuadro 7-2 – ACF[0] para la aplicación de la Escalera de Claves

Acf[0]	Valor	Descripción
AcfCw1Mode	0x11	Operación **Escalera de Claves** tal como se define en la presente Recomendación. acf [1]..acf[14] será 0x00.
AcfAk1Mode	0x12	Operación del **Mecanismo de Autenticación** tal como se define en la presente Recomendación. Al valor de acf[1] se hace referencia como AkModeField. Los valores aplicables se definen en el Cuadro 7-3. Acf[2]..acf[14] será 0x00.
reservado	Otro	Reservado para uso futuro

La definición suplementaria en notación C de este AcfCw1Mode para su aplicación como parámetro de la **Escalera de Claves** es:

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

La definición suplementaria en notación C del modo AcfAk1Mode para su aplicación como parámetro de la **Escalera de Claves** es:

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

Cuadro 7-3 – Definición de AkModeField para AcfAk1Mode

Registro	Bit	Valor		Descripción
AkUseFlag	8	0b0	**AkUseAS**	Aplicación de AK exclusivamente para el **Sistema de Seguridad Avanzada**.
		0b1	**AkUseCI**	Aplicación de AK para el **Cliente ECI**.
AkOnline	7	0b0	**AkOffline**	Se establece AK en modo "fuera de línea" unidireccional. Las puestas a prueba/contestaciones pueden precalcularse.
		0b1	**AkOnline**	Se establece AK utilizando una AKRK de palabra de ocasión ("nonce") aleatoria que exige una puesta a prueba/contestación calculada "en línea".
AkAsAppl solo si AkUseFlag= **AkUseAS** En otro caso queda en reserva	0..3	0x0	**AkConfigAuth**	Autentica el elemento de Configuración del **Intervalo AS**.
		0x1	**AkLdUssk**	Utilización de AK para desencriptar y cargar una clave uSSK de **Microservidor**.
		0x2	**AkCIImg**	Utilización de AK para desencriptar la clave que permite desencriptar la imagen de **Cliente ECI** que se va a cargar.
		0x3.. 0xF	Reservado	Reservado para uso futuro.
RFU	Otros	0		Reservado para uso futuro. El valor se pondrá a cero.

8 Intervalo de Seguridad Avanzada

8.1 Introducción al Intervalo de Seguridad Avanzada

El **Sistema de Seguridad Avanzada** contiene información del estado de cada **Cliente ECI** cargado. La identificación que vincula un **Cliente ECI** con un **Intervalo AS** es la clave pública de **Operación de Plataforma** (POPK) del **Cliente ECI**. El **Anfitrión ECI** carga la POPK de un **Cliente ECI** en un **Intervalo AS** disponible. A partir de ese momento, el estado del **Intervalo AS** se asocia con ese **Cliente ECI**. Cualquier operación relevante del **Intervalo AS** utilizará la POPK como dato de entrada, haciendo que el resultado sea específico para el **Cliente ECI** vinculado y que carezca de sentido para los demás.

La **Robustez** del **Anfitrión ECI** garantiza que un **Cliente ECI** sólo pueda acceder a información del intervalo asignado al mismo: si el **Anfitrión ECI** funciona correctamente, asegurará que sólo el **Cliente ECI** designado tenga acceso a su propio **Intervalo AS**. En caso de que el **Anfitrión ECI** se vea comprometido de alguna forma, el "mecanismo de bloqueo" de la POPK garantiza que sólo las entradas AS a un **Intervalo AS** que constituyan un conjunto consistente puedan dar lugar a resultados que tengan sentido en forma de claves de desencriptación (CW) y **Propiedades del contenido** asociadas o una Clave de Autenticación (AK).

Si el **Anfitrión ECI** decidiera replantear el uso de un **Intervalo A**S para otro **Cliente ECI**, se borran todos los estados del **Cliente ECI** previo (POPK).

8.2 Definición del Intervalo AS

8.2.1 Generalidades

Un **Intervalo AS** se define en términos de la relación de ***variables de estado*** y ***variables de entrada*** con ***funciones de modificación de estado***. La representación que en esta cláusula se hace del estado y de los valores de entrada y salida, se ha elegido de forma que las operaciones sobre los mismos se definen mediante representaciones binarias aquí definidas. Esto es particularmente pertinente en relación con su inclusión en cálculos criptográficos y de la **Escalera de Claves**. Las implementaciones reales pueden seleccionar sus propias representaciones de estado pero deben traducir cualquier representación de cliente a una representación aquí especificada como entrada a cualquier operación criptográfica.

Todas las variables de entrada de un **Intervalo AS** estarán protegidas de forma robusta contra cualquier modificación maliciosa. Algunas variables de estado contienen información que debe permanecer secreta: esos registros estarán protegidos de forma robusta contra accesos no autorizados. Esas variables se definen utilizando "Secreto" como parte de la definición del tipo en notación C. Cualquier cálculo basado en el valor de una variable secreta se mantendrá secreto excepto cuando dicho resultado sea compartido de forma explícita. Toda ubicación de almacenamiento de un valor secreto y de un cálculo con un valor secreto tendrá la misma **Robustez** que la necesaria para el **Bloque de Escalera de Claves** [UIT-T J.1015].

El **Intervalo AS** consta de sesiones. Cada sesión opera conforme a los valores establecidos en su configuración, que forma parte del estado de la sesión. El **Cliente ECI** establece la configuración de la sesión, que debe ser autenticada antes de su uso mediante el **Mecanismo de Autenticación** o mediante las propiedades de autenticación implícita de la **Escalera de Claves**.

Todas las variables de estado y las funciones se definen con la notación del lenguaje C [ISO/CEI 9899]. El orden secuencial del lenguaje C no se observa estrictamente en el sentido de que los elementos pueden definirse después de su uso. Las matrices de tamaño fijo se copian con una única declaración de asignación (en lugar de copiar el valor del puntero), como si estuvieran contenidas en una estructura "struct".

Con relación a los errores, el código es más legible en virtud de la definición de una verificación implícita de errores. Asignar un valor reservado a una variable de estado o campo de la misma (véase la definición de campo) da lugar a un error. Si el término a la derecha de esa expresión de asignación se basa en un único parámetro de función, se devuelve un error para ese parámetro: valor -i para el parámetro i ("value -i for parameter i").

Todos los valores por defecto de campos y variables se definen como 0, salvo que se especifique otra cosa.

8.2.2 Definición del estado del Intervalo AS

8.2.2.1 Estado del intervalo y la sesión

El estado del **Intervalo AS** se define como un estado común del intervalo y un estado de sesión para cada sesión de encriptación y desencriptación. La estructura del estado de intervalo se define a continuación. Los campos se definen en el Cuadro 8-1.

```
#define NSLOTS       /* (máximo) número de intervalos */
#define NSESSIONS    /* (máximo) número de sesiones */
#define MaxSpkEncr 4 /* número máximo de valores de cifrado SPK */

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]
```

Cuadro 8-1 – Definición de la estructura del estado del Intervalo AS

Campo	Descripción
active	Es verdadero si la sesión está activa, falso en caso contrario. El estado por defecto es Falso.
configAuthMode	Modo de autenticación de la configuración del intervalo. Los valores permitidos son: **ConfigAuthModeNone**: 0x0, la configuración del intervalo no ha sido autenticada. **ConfigAuthModeAk1**: 0x1, la configuración del intervalo ha sido autenticada mediante el mecanismo AK definido en la cláusula 8.2.4.8. Todos los demás valores están reservados.
Mh	Asa de medios a la que se asocia el **Intervalo AS**.
clientCheckFlag	Se ha cargado un nuevo Cliente ECI. La verificación de **POClRLVnr** se realizará durante la inicialización de la sesión. El valor por defecto es 0b1.
Reservado	Campo reservado; se pondrá a cero.
POClRLVnr	Número de versión de la **Lista de Revocación** del cliente **Operación de Plataforma** que ha sido utilizada para verificar el **Cliente ECI** antes de la carga. En cada inicialización de sesión de **Cliente ECI** se verificará con respecto a la versión mínima que este espera encontrar.
slotConfig	Configuración del intervalo.
spk	Clave pública utilizada para calcular LK1 y AK.
spkUri	Registro de información de reglas de uso del vector SPK utilizado para calcular LK1.
spkIndx	Índice de selección de la ubicación del registro de SPK en el vector SPK para el cálculo de LK1.
coupledSessionId	Sólo es aplicable si la sesión del **Intervalo AS** está en modo desencriptación. Segunda sesión (desencriptación) emparejada con esta. Los flujos de contenido decodificados se agrupan y se comparan las **Propiedades del Contenido**. El valor por defecto es –1.
nEncr	Número de valores de entrada de SPK/POPK utilizados en la encriptación (se excluye el spk del intervalo).
encrSpk	Valores de SPK para la encriptación de contenido con la **Escalera de Claves**.
encrPopk	Valores de POPK para la encriptación de contenido con la **Escalera de Claves**.
encrCwUri	Valores de CwUri para la encriptación de contenido con la **Escalera de Claves**.
lk1	Clave de enlace de nivel superior utilizada para el cálculo de palabras de control con la **Escalera de Claves**.
Ussk	Clave secreta del **Microservidor** (para aplicaciones del **Microservidor**).
rkState	Estado de la clave aleatoria de la sesión.
ies	Estado de importación/exportación de la sesión.
version	Versión del estado del intervalo. Los valores permitido son: 0x1: versión 1. Todos los demás valores están reservados.
slotMode	Modo en el que funciona el intervalo. Los valores permitidos son: **SlotModeDecr**: 0x1, el intervalo funciona en modo desencriptación. **SlotModeEncr**: 0x2, el intervalo funciona en modo encriptación. Todos los demás valores están reservados.
popk	Clave pública del **Cliente ECI** que utiliza este intervalo.
slotRK	Número aleatorio utilizado en los protocolos en línea de contestación a puestas a prueba, por ejemplo, con un **Servidor de Aprovisionamiento**. El valor se fija durante la inicialización del intervalo.
akClient	Clave de autenticación con fines de procesamiento del cliente.
se	Estado de sesión (para todas las sesiones de un **Intervalo AS**).
ss	Estado de **Intervalo AS** (para todos lo intervalos).

Salvo que se especifique otra cosa, el valor por defecto de cada elemento de estado en la inicialización es cero.

8.2.2.2 Configuración de la desencriptación

El estado de la configuración aplicable a una sesión de **Intervalo AS** en modo desencriptación se define en notación en lenguaje C más abajo indicada y se describe en el Cuadro 8-2. Define la información detallada de la operación de sesión en el Intervalo AS cuando esta se encuentra en modo desencriptación. Estos datos pueden autenticarse mediante la aplicación del **Mecanismo de Autenticación** o los cálculos de la **Escalera de Claves** adecuados.

```
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;
```

Cuadro 8-2 – Definición de la estructura de DecryptConfig

Campo	Descripción
configVersion	Versión de la configuración de desencriptación. El valor definido es 0x1. Todos los demás valores están reservados. Una sesión **del Intervalo AS** rechazará ejecutar cualquier función de transición de estado si este campo no tiene un valor permitido.
reserved1	Campo reservado: se pondrá a 0.
klModeAuth	Si este bit tiene el valor 1, la sesión del **Intervalo AS** aplicará ClientConfig para la autenticación con los cálculos de la **Escalera de Claves**. Este bit se autentica en todos los cálculos de **Escalera de Claves**.
akModeAuth	Si este bit tiene el valor 1, la sesión del **Intervalo AS** verificará que configAuthMode toma el valor ConfigAuthModeAk1 antes de permitir ningún cálculo de **Escalera de Claves**. Este bit se autentica en todos los cálculos de **Escalera de Claves**.
rkKlMode	Si esta bandera tiene el valor 1, se aplicará **slotRK** en todos los cálculos de **Escalera de Claves** para el **Intervalo AS**.
spk0NoDecrypt	Cuando se pone a 1 no está permitido utilizar spk[0] (spkIndx==0 como entrada a la función Escalera de Claves) para la autenticación de la LK1 del intervalo en modo desencriptación.
reserved2	Campo reservado: se pondrá a 0.
rkDecrMode	Define la aplicación de una clave de sesión aleatoria para los cálculos de Escalera de Claves. Véase la cláusula 8.2.2.5.
minEciRootState	Valor mínimo para la versión de raíz **ECI** y la versión de la **Lista de Revocación** raíz. Si el CPS ha aplicado una **Clave Raíz ECI** o **una Lista de Revocación** raíz con fines de autenticación **ECI** de valor inferior a los valores de esta estructura, no se permitirá realizar ningún cálculo de Escalera de Claves en esa sesión.
minClientVersion	Versión del **Cliente ECI**. Se utiliza para verificar los números de versión de la **Lista de Revocación** para POPK.

8.2.2.3 Configuración de la encriptación

El estado de la configuración aplicable a una sesión de **Intervalo AS** en modo encriptación se define mediante la notación en lenguaje C abajo indicada y se describe en el Cuadro 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;
```

Cuadro 8-3 – Definición de la estructura de EncryptConfig

Campo	Descripción
configVersion	Versión de la configuración de encriptación. El valor definido es 0x1. Todos los demás valores están reservados. Una sesión del **Intervalo AS** rechazará la ejecución de cualquier función de transición de estado si este campo no tiene un valor permitido.
Reserved1	Campo reservado: se pondrá a 0.
microServerVersion	Número de versión de la configuración del **Microservidor**. También se utiliza como número mínimo de versión de la **Lista de Revocación** para la autenticación del **Microcliente** en modo **Microservidor** asimétrico.
asymKlMode	Si esta bandera tiene el valor 1, la **Escalera de Claves** funcionará de conformidad con el **Mecanismo de Autenticación** de cliente asimétrico definido en la cláusula 7.3.
rkKlMode	Si esta bandera tiene el valor 1, se aplicará slotRK en todo cálculo de la **Escalera de Claves**.
Reserved2	Campo reservado: se pondrá a 0.
rkEncrMode	Define la aplicación de una clave aleatoria para los cálculos de **Escalera de Claves**. Véase la cláusula 8.2.2.5.
basicUriTrfr	Define las transformaciones de estado de la URI básica desde la conexión de importación antes de aplicar la URI básica como propiedad del contenido del contenido encriptado. Véanse en el Cuadro 8-5 los correspondientes valores.
contPropControl	Define cómo se calculan las **Propiedades del Contenido** del contenido encriptado. Véase el Cuadro 8-5.
defaultCP	Valor por defecto para todos los campos propiedades del contenido. Su aplicación en el cálculo de la **Escalera de Claves** está bajo control del campo contPropControl.
minEciRootState	Valor mínimo para la versión de raíz **ECI** y la versión de la **Lista de Revocación** raíz. Si el CPS ha aplicado una **Clave Raíz ECI** o una **Lista de Revocación** raíz con fines de autenticación **ECI** con un valor inferior a los valores de esta estructura, no se permitirá ningún cálculo de **Escalera de Claves**.

El campo contPropControlFields es una matriz de 16 campos de 2 bits. Los campos de 2 bits indican cómo se controlan las **Propiedades del Contenido** de Field1 para la salida encriptada. En el Cuadro 8-4 se muestra la información relativa a la desencriptación. Los bits 2n y 2n+1 de CpControlFlag se corresponden con el byte n de Field1.

Cuadro 8-4 – Definición de CpCtrl

Nombre de la bandera	Valor	Descripción
CpCtrlCopy	0b00	El byte propiedades del contenido CP Field1 se copiará de una conexión de importación.
CpCtrlDef	0b01	El byte propiedad del contenido CP Field1 tomará el valor del correspondiente byte defaultCP.
CpCtrlMS	0b10	El byte propiedad del contenido CP Field1 es fijado por el **Microservidor**.
Reservado	0b11	El valor queda reservado.

El campo basicUriTrfr modifica el comportamiento antes descrito de CPControlFlags para el campo BasicUri cuando el estado de su CPControlFlag es el mismo que el de **CPCopy**. En el Cuadro 8-5 se define el comportamiento alternativo.

Cuadro 8-5 – Valores y descripción de BasicUriTrfr

Nombre de la bandera	Valor	Descripción
JustCopy	0x00	El byte propiedades del contenido de Field1 se copia de la conexión de importación.
NoMoreCopy	0x01	Un estado basicURI de RedistributionProtected pasará al estado **ViewOnly**.
Reservado	Otros	Reservado para uso futuro.
NOTA – Al fijar que el estado BasicUriTrfr sea NoMoreCopy, el sistema del **Microcliente** sólo permitirá el establecimiento de un flujo en cualquier entrada de contenido protegido al **Microservidor**.		

8.2.2.4 Control de la clave de sesión aleatoria

La estructura RKMode en notación C descrita a continuación y en el Cuadro 8-6 define la forma de aplicar la clave de sesión aleatoria en la **Escalera de Claves**.

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

Cuadro 8-6 – Estructura de la clave aleatoria para una sesión de desencriptación y encriptación

Campo	Descripción
mode	Define el modo de aplicación de la clave de sesión aleatoria. Los valores son los siguientes: • 0b00: **RKModeNone**, no se inserta ninguna clave de sesión aleatoria. • 0b10: **RKModeDataLimit** se aplica una clave de sesión aleatoria con limitación en los datos. • 0x11: **RKModeTimeLimit** se aplica una clave de sesión aleatoria con limitación en el tiempo. • 0b01: valor reservado.
limit	El valor define el límite aplicable expresado en segundos en tiempo real o en Kbytes de datos desencriptados o encriptados desde la inicialización de la clave aleatoria. La función limitValue() define el valor límite actual aplicable. El valor 63 queda reservado.

```
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 Configuración completa de la sesión

La configuración completa de la información de control para una sesión de **Intervalo AS** en modo encriptación o desencriptación se define en la estructura SesionConfig que se muestra a continuación. Si el **Intervalo AS** se encuentra en modo encriptación incluye la información de configuración para la subsiguiente desencriptación.

```
typedef struct SessionConfig {
    EncryptConfig   encryptConfig; /* configuración para encriptación */
    DecryptConfig   decryptConfig; /* configuración para desencriptación */
} SessionConfig;
```

La estructura cpsEciRootState que define el Estado Raíz **ECI** para validar **Cadenas de Certificados ECI** se define en la notación C que se indica más abajo y en el Cuadro 8-7.

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

EciRootState cpsEciRootState; /* contiene el valor mínimo del CPS */
```

Cuadro 8-7 – Descripción de campos de la estructura EciRootState

Campo	Descripción
rootVersion	Versión del **Certificado Raíz ECI**.
rlVersion	Versión de la **Lista de Revocación** aplicada con el **Certificado Raíz**.
NOTA – EciRootState se aplica normalmente como una limitación inferior (valor mínimo) permitido para la versión raíz **ECI** y la versión de la **Lista de Revocación** cuando se carga la información del **Anfitrión ECI** y el **Cliente ECI**.	

La función siguiente verifica si cpsEciRootState es suficiente para el cálculo de una clave:

```
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);

  /* lo siguiente no debería producirse */
  return false;
}
```

Precondiciones:

- Se ha inicializado el SlotId del **Intervalo AS**.

8.2.2.6 Estado de la clave de sesión aleatoria

Para cada sesión de desencriptación o encriptación asociada a un **Intervalo AS**, el **Intervalo AS** almacena información del estado de la clave aleatoria tal como se define a continuación en notación C descrita en el Cuadro 8-8.

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

Cuadro 8-8 – Descripción del campo estado de la clave aleatoria RkState

Campo	Descripción
rkCurrent	Variable aleatoria actual utilizada para su inserción en la **Escalera de Claves** para el cálculo de CW.
rkNext	Siguiente valor de la clave aleatoria a insertar en la **Escalera de Claves** para el cálculo de CW.
limitCounter	Contador que indica el estado de utilización de la clave actual en unidades relacionadas con el valor límite que se aplica a la clave. El valor cuenta las unidades pendientes que aún pueden ser encriptadas o desencriptadas con una CW calculada con rkCurrent.

El campo **limitCounter** se incrementará al aplicar la CW.

NOTA – Las implementaciones puede incluir el contador como parte del **Trayecto de Vídeo Seguro**.

8.2.2.7 Estado de importación y exportación

Cada sesión de encriptación tiene una sesión de desencriptación asociada de la que importa el contenido a reencriptar. Es posible realizar simultáneamente la importación de (al menos) dos grupos de exportación de las sesiones exportadoras, permitiendo así un traspaso sin solución de continuidad.

Es posible emparejar dos sesiones de desencriptación. Ello permite fusionar distintos subflujos que necesitan palabras de control distintas (calculadas para el mismo **Intervalo AS**) en un flujo compuesto con un conjunto de **Propiedades del Contenido** antes de su envío a salidas normalizadas de la industria o exportaciones. Como parte de la fusión, el **Sistema AS** verifica que los flujos fusionados tienen las mismas **Propiedades del Contenido**.

NOTA – La comparación de las **Propiedades del Contenido** también puede incluir el id del grupo de exportación, lo que garantiza que la necesidad del procesamiento de la cadena de exportación para ambas sesiones agrupadas es igual para ambos casos.

Junto con el estado de la clave aleatoria, este es el estado de vinculación de sesión en el **Intervalo AS**. A continuación se define mediante código en notación C el estado de sesión "se"; la descripción de cada campo figura en el Cuadro 8-9.

```
#define MaxExpGroupIds 2

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

#define ImportNone -1
```

Cuadro 8-9 – Definición de la estructura de ImportExportState

Campo	Descripción
importSlotId	Sólo es aplicable si el **Intervalo AS** está en modo encriptación. El valor es el número de intervalo del que se importa el contenido ('intervalo de importación'). El valor por defecto es –1.
importSession	Sólo es aplicable si el **Intervalo AS** está en modo encriptación. El valor es el número de sesión en el intervalo de importación del que se importa el contenido. El valor por defecto es –1.
expGrpId[eid]	Sólo es aplicable si el **Intervalo AS** está en modo encriptación. Es el id del grupo de exportación del **Intervalo AS** exportador del que puede importarse el contenido. El valor 0x00 está reservado.
importPermitted[eid]	Sólo es aplicable si el **Intervalo AS** está en modo encriptación. Toma el valor verdadero si el **Intervalo AS** exportador permite el valor de **expGrpId[eid]**; falso en caso contrario. El valor por defecto es falso.
rkState	Estado de la clave de sesión aleatoria para esta sesión.

El **Sistema AS** cancelará una sesión de importación (fijando en falso el correspondiente campo ImportPermitted) si se reinicia la sesión de desencriptación. El **Sistema AS** reiniciará todas las sesiones de un **Intervalo AS** cuando se produzca la reinicialización o puesta al valor de fábrica de un **Intervalo AS**.

8.2.3 Autenticación de las Propiedades del Contenido

Los **Clientes ECI** que realizan funciones de desencriptación proporcionan al **Anfitrión ECI** los valores de las propiedades del contenido a través de la correspondiente API de propiedades del contenido. El **Anfitrión ECI** proporcionará esos valores al **Sistema de Seguridad Avanzada** junto con los datos necesarios para calcular la palabra de control para el contenido aplicable. El **Sistema de Seguridad Avanzada** garantizará la aplicación adecuada de las **Propiedades del contenido** y validará las **Propiedades del contenido** utilizándolas en el cálculo del valor de C-input para el **Bloque de Escalera de Claves**.

Los **Microservidores** utilizan las **Propiedades del contenido** que pasa o procesa el **Sistema AS** o el **Cliente ECI** utilizando el mismo mecanismo arriba indicado en el cálculo del valor de C-input para la **Escalera de Claves**. Los **Intervalos AS** utilizados en el modo encriptación cotejan las **Propiedades del Contenido** proporcionados por el **Microservidor** con las enviadas por el recurso de desencriptación con arreglo a los valores de configuración del **Microservidor**. Cuando se detecta una falta de concordancia se detiene la encriptación.

Para fines de autenticación y verificación, las **Propiedades del Contenido** se combinan en una secuencia de bytes en dos fases. En la primera se combinan campos más pequeños de longitud fija de la propiedad del contenido en el *field1*. El byte fieldControl controla la presencia de campos de propiedades del contenido del tamaño de un byte con fines de autenticación. En la segunda fase, los campos de propiedad del contenido más extensos se combinan en una secuencia de bytes *field2*. *Field1* y *field2* se concatenan y constituyen la entrada a una función *hash* que condensa todos los

campos en un valor de 128 bits para C-input de la **Escalera de Claves**. En el Cuadro 8-10 se describe la estructura de field1.

Cuadro 8-10 – Definición de la estructura de field1

Nombre	Tipo	Número de bytes	Descripción
fieldControl	FieldControl	0,1	Este campo define un valor de 16 bits, con los bits menos significativos en el byte 0. Véase el Cuadro 8-11.
basicUri	byte	2	El valor de este campo corresponderá a la especificación del tipo BasicUri, [UIT-T J.1012], Cuadro 9.8.2.5.1-1.
outputControl	byte [2]	3-4	El valor de este campo corresponderá a la especificación del Vector de Control de Salida [UIT-T J.1012], Cuadro 9.8.2.6.1-1.
standardUri	byte [3]	5-7	El valor de este campo corresponderá a la especificación de la URI estándar [UIT-T J.1012], Cuadro 9.8.2.3.1-1.
exportGroup	byte	8	Se interpreta como un entero sin signo que representa el id del grupo de exportación que se aplica al contenido. **El Anfitrión ECI** interpretará el valor 0 como que no se permite exportación alguna; los valores 0x80 – 0xFF están reservados.
parentalAuth	byte	9	Se corresponde con ParCond.basicCondition definido en [UIT-T J.1012], cláusula 9.8.1.7.7-1, con los bits [0..5] puestos a 0b000000.
Reservado	byte[6]	10-15	Los Anfitriones ECI conformes con esta Recomendación pondrán estos bytes a 0x00.

Cuadro 8-11 – Definición de la estructura de FieldControl

Nombre	Bit(s)	Descripción
bit-<n>	2-16	Este bit controla la validación del byte -<n> del campo field1. Si el valor es 0b1 indica que el valor del byte -<n> será validado e igual al campo indicado, si el valor es 0b0 indicará que el byte -<n> no será validado y en su lugar se usará el valor 0x00 para el byte-<n> en field1. Bit-2 se pondrá a 0b1 cuando se utilice como entrada para calcular una palabra de control de desencriptación. Eso garantiza que basicUri siempre sea autenticado contra el valor usado en el momento de la encriptación del contenido.
Field2ctrl	0-1	El valor 0b00 indica que field2 no está presente. El valor 0b01 indica que el campo está presente y utiliza la codificación tal como se define más abajo. Los valores 0b10 y 0b11 están reservados y no se utilizarán.

La definición de Field2 utiliza una estructura etiqueta/longitud/valor con una longitud total del campo que garantice la integridad del conjunto. La estructura de Field2 se define más adelante en esta cláusula.

La función computeField1Decrypt calcula la lógica de selección para el paso siguiente de la autenticación:

```
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;
}
```

Un **Intervalo AS** en modo encriptación calculará la entrada para la autenticación de la propiedad del contenido denominada result1 y un campo cpMask para comparar los bytes de field1 con el field1 perteneciente al contenido de la sesión importadora de la forma siguiente:

```
    void computeField1Encrypt(
                uchar msField1[16],/* field1 para CP del Cliente Microservidor */
result1[16],            /* CP resultado para la autenticación en el cálculo de CW */
                ushort cpMask,              /* máscara para comparar msField1 con la
                                               versión de field1 de CP del cliente */
                EncryptConfig ssEncrypt /* configuración de la encriptación del intervalo AS */
    ) {
        int i;
        uchar cp[16]; /* valor de CP a calcular */

        /* fija los bytes de control de las propiedades del contenido */
        cp[0] = ssEncrypt.defaultCp[0];
        cp[1] = ssEncrypt.defaultCp[1];
        mask = 0x0000;

        /* procesa las reglas de contPropControl para calcular cp */
        for (i=2; i<16; i++) {
            switch (ssEncrypt.contPropControl>>(2*i) && 0b11) {
                case CpCtrlCopy: /* se copiará del Cliente de importación */
                    if (i==2) { /* byte de la URI básica */
                        /* procesa 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 {   /* todos los demás bytes de CP */
                        cp[i]= msField1[i];
                    }
                    cpMask += 1<<i; /* byte I de msField1 que se compara con el cliente de imp */
.                   break ;
                case CpCtrlDef: /* tomará el valor del CP por defecto de la configuración */
                    cp[i] = ssEncrypt.defaultCP[i] ;
                    break ;
                case CpCtrlMS: /* se definirá por el software del Microcliente */
                    cp[i] = msField1[i];
                    break;
            }
        }

       /* calcula la entrada a la función de autenticación de la misma forma que para la
       desencriptación */
        computeField1Decrypt(cp, result1);
    }
```

Field2 es una secuencia estructurada de bytes tal como se define a continuación:

```
typdef struct Field2 {
    uint length;        /* número de bytes del contenido, será múltiplo de 4 */
    byte content[]; /* contenido definido más abajo */
} Field2;
```

El campo de contenido de la estructura Field2 contendrá una secuencia de estructuras LargeProperty cada una con una etiqueta única. LargeProperty se define mediante el código siguiente en notación C:

```
typdef struct LargeProperty {
    uint  propertyTag;  /* véase el Cuadro 8-12  */
    uint  length;       /* longitud del campo property en bytes
    byte  property[];   /* contiene el valor real de property */
    byte  padding[];    /* bytes adicionales puestos a 0x00 para que el tamaño de LargeProperty sea
                           múltiplo de 4 bytes */
} LargeProperty;
```

Los valores del campo propertyTag de largeProperty y las correspondientes definiciones del campo de propiedades se muestran en el Cuadro 8-12.

Cuadro 8-12 – Valores y significado del campo etiqueta de largeProperty

Valor de propertyTag	Propiedad (Property)
0x00000000	Reservado.
0x00000001	Corresponde a **datos** de parámetros del mensaje **setDcrMarkBasic** tal como se define en [UIT-T J.1012], cláusula 9.8.2.7.5.
0x00000002	Corresponde a **datos** de parámetros del mensaje **setDcrMarkExt** tal como se define en [UIT-T J.1012], cláusula 9.8.2.7.6.
0x00000003	Corresponde al parámetro **custURI** del mensaje **setDcrCustUri** tal como se define en [UIT-T J.1012], cláusula 9.8.2.4.1.
Otros	Reservado para uso futuro.

El **Sistema AS** puede rechazar aquellos datos que excedan su capacidad de proceso para *field2*.

El **Sistema AS** verificará la coherencia de cualquier parámetro de datos de Field2 mediante las comprobaciones siguientes:

- Longitud de las estructuras LargeProperty constituyentes igual al campo longitud de la estructura Field2.
- Los bytes de relleno de todas las estructuras LargeProperty constituyentes son 0x00.

El input-C de los datos asociados para la **Escalera de Claves** se calculará a partir de result1 y field2 conforme al código siguiente en notación 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) {
        /* no se incluirá ningún field2 */
        for (i=0; i<16; i++) hashIn[i] = result1[i];
        asHash(hashIn, 16, 128, input_C);
    } else if result[0] & 0b11 = 0x01) {
        /* field2 se incluirá para el 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 es la función hash definida en la cláusula A.1 de una secuencia de bytes en el primer parámetro, la longitud de la secuencia de bytes en el segundo parámetro, la longitud en bits del resultado en el tercer parámetro y el resultado en el último parámetro.

La **Robustez** del cálculo de la Hash exterior (el cálculo directo será al menos tan potente como el cálculo Hash de la función Hash interior). La medida de la **Robustez** de un hash refleja el esfuerzo necesario para crear una discrepancia entre cualquiera de las entradas de la función hash y la aplicación de esas entradas como propiedades del contenido así como la manipulación de la función hash y/o sus entradas.

Un ejemplo de los distintos niveles de **Robustez** de los dos cálculos hash es que el hash exterior puede realizarse con un bloque hardware robusto mientras que el hash interior puede realizarse mediante una implementación robusta del software.

8.2.4 Funciones del Intervalo AS

8.2.4.1 Visión general

El **Sistema AS** puede realizar varias funciones en nombre de los **Clientes ECI** a través del **Anfitrión ECI**. Estas funciones constituyen la base de la API de Seguridad Avanzada descrita en [UIT-T J.1012]. Un "evento" permite informar a un **Cliente ECI** de la ocurrencia de un evento asíncrono. No está previsto poder contestar al mismo. Todas las demás funciones se consideran mensajes asíncronos o mensajes síncronos iniciados por el **Cliente ECI**; sus valores de retorno indican el estado de la contestación. Las funciones figuran en el Cuadro 8-13.

Cuadro 8-13 – Visión general de las funciones de Seguridad Avanzada

Nombre de la función	Descripción	Cláusula
reqAsInitSlot	Inicializa un **Intervalo AS**.	8.2.4.2
reqAsAStartDecryptSession	Inicia una sesión de desencriptación en un **Intervalo AS**.	8.2.4.3
reqAsCoupleDecryptSession	Agrupa dos sesiones de desencriptación en una sola.	8.2.4.3
reqAsDecoupleDecryptSession	Desagrupa dos sesiones de desencriptación previamente agrupadas.	8.2.4.3
reqAsStartEncryptSession	Inicia una sesión de encriptación.	8.2.4.3
callAsNextKeySession	Cambia a la siguiente clave aleatoria.	8.2.4.3
reqAsStopSession	Detiene una sesión.	8.2.4.3
reqAsExportConnSetup	Establece una **Conexión de Exportación** desde una sesión de desencriptación a encriptación.	8.2.4.4
reqAsExportConnEnd	Termina la sesión de exportación existente.	8.2.4.4
reqAsLoadLk1	Carga la clave de enlace de nivel superior en la **Escalera de Claves** de una sesión.	8.2.4.5
reqAsComputeEncrCw	Calcula la palabra de control de encriptación.	8.2.4.6
reqAsComputeDecrCw	Calcula la palabra de control de desencriptación.	8.2.4.7
reqAsComputeAkClient	Calcula la clave de autenticación a utilizar por el **Cliente ECI**.	8.2.4.8
reqAsClientChalResp	Uso de la clave de autenticación en nombre del **Cliente ECI**.	8.2.4.8
reqAsAuthDecrConfig	Autentica la configuración de la sesión mediante **Mecanismos de Autenticación** (modo desencriptación).	8.2.4.9
reqAsAuthEncrConfig	Autentica la configuración de la sesión y los parámetros de encriptación **mediante Mecanismos de Autenticación** (modo encriptación).	8.2.4.9
reqAsLdUssk	Carga la clave secreta del **Microservidor**.	8.2.4.10
reqAsMInikLk1	Calcula el mensaje de inicialización asimétrica del **Microcliente**.	8.2.4.11
reqAsClientImageDecrKey	Calcula la clave de desencriptación de la imagen del **Cliente ECI**.	8.2.4.12
getAsSlotRk	Lee la clave aleatoria del intervalo.	8.2.4.13
getAsSessionRk	Lee la clave aleatoria de la sesión.	8.2.4.13
getAsSessionLimitCounter	Lee el resto de las unidades de la clave aleatoria de la sesión.	8.2.4.13
setAsSessionLimitEvent	Se envía evento cuando se alcanza el valor límite del resto de las unidades.	8.2.4.13
reqAsEventSessionLimit	Mensaje de evento cuando se ha alcanzado el valor límite.	8.2.4.13
getAsClientRnd	Obtiene un nuevo número aleatorio para las aplicaciones del **Cliente ECI**.	8.2.4.13
getAsSC	Obtiene el estado del campo control de aleatorización actual del contenido en una sesión.	9.9
reqAsEventCpChange	Mensaje de evento cuando se modifican las propiedades del contenido del contenido importando en una sesión de encriptación.	9.9
setAsPermitCPChange	Habilita/deshabilita que los cambios en las propiedades del contenido (CP) realizadas afecten a la selección de la palabra de control para encriptación en una sesión de encriptación.	9.9
setAsSC	Fija el campo control de aleatorización del contenido encriptado en una sesión de encriptación.	9.9
reqAsEventSC	Mensaje de evento cuando se modifica el campo control de aleatorización en una sesión.	9.9

El pseudocódigo de las subcláusulas de la presente cláusula contiene códigos de error como valores de retorno de las funciones. Los valores del código de error se definen en la cláusula 8.2.4.15 incluyendo una descripción de los mismos.

8.2.4.2 Inicialización del Intervalo AS

Al realizar la carga, el **Anfitrión ECI** reserva un **Intervalo AS** en el **Sistema de Seguridad Avanzada** en nombre del **Cliente ECI** que se va a cargar. El **Anfitrión ECI** invocará la función reqAsInitSlot definida más abajo. Toda la información de estado del **Intervalo AS** se fijará en su estado por defecto; toda **Conexión de Exportación** será reinicializada. El **Anfitrión ECI** cargará el **Cliente ECI** mediante el núcleo o elementos fundamentales del cargador (véase la cláusula 11). El *POClRLVnr* del **Intervalo AS** *reflejará el número mínimo de versión de la Lista de Revocación POC utilizado para validar la imagen del cliente.* Ese valor será verificado cuando el **Cliente ECI** inicie una sesión.

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

Semántica:

Toda la información de estado de **slotId** del **Intervalo AS** tomará el valor por defecto; se reinicializarán todas las **Conexiones de Exportación**.

La carga de POPK precisa del suministro de la cadena para el procesamiento del Sistema CPS. Las reglas para el procesamiento de POPK se definen en la cláusula 10.4. ECI_Certificate_Chain se define en la cláusula 5.4.1 de [UIT-T J.1012]. Una vez validado satisfactoriamente se ejecutará el siguiente código en notación C:

```
/* inicializa el estado del intervalo */
ss[slotId].popk = /* valor validado de popk devuelto por CPS */;
ss[slotId].POClRLVnr = /* valor utilizado para la verificación de la imagen del cliente*/;
ss[slotId].version = slotVersion;
ss[slotId].slotMode = slotMode;
ss[slotId].configAuthMode = ConfigAuthModeNone;
ss[slotId].rkSlot = rnd128();
return ErrOk;
```

La función rnd128() devuelve un número aleatorio de 128 bits, tal como se define en la cláusula A.3 como una matriz compuesta por 16 uchar.

8.2.4.3 Control de la sesión y la clave aleatoria del Intervalo AS

Un **Intervalo AS** permite varios estados de sesión para diferentes sesiones simultáneas. La función siguiente inicia y detiene sesiones en un intervalo:

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

Semántica:

Se ejecutará el siguiente código en notación C:

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

/* verifica si se ha utilizado una lista de revocación de cliente válida */
if (config.decryptConfig.clientVersion >
    ss[slotId].clientPOClRLVnr) return ErrRevocEnforce;

/* localiza cualquier sessionId disponible; todo algoritmo es ok */
int i=0;
while (i<NSESSIONS && ss[slotId].se[i].active) i++;
if (i==NSESSIONS) return ErrNoMoreSessions;
/* i contiene un bloque de administración de sesión inactivo */
*sessionId = i;
```

```
/* inicializa el estado de la sesión */
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;
```

Precondiciones:

- El **Intervalo AS** ha sido inicializado satisfactoriamente.

NOTA – El parámetro mh (asa de medios) permite al **Anfitrión ECI** identificar la sesión de desencriptación AS asociada a la sesión de desencriptación del contenido que ha iniciado. No lo utiliza el **Sistema AS**.

La función coupleDecrypSessions permite agrupar dos sesiones inicializadas. La segunda sesión se acopla a la primera; la primera se convierte en el asa principal a efectos del contenido combinado.

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

Semántica:

Se ejecutará el siguiente código en notación 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;
/* se informa al trayecto de vídeo seguro de la agrupación de sesiones &/

return ErrOk;
```

Precondiciones:

- Ambas sesiones del **Intervalo AS** han sido iniciadas satisfactoriamente.

Puede hacerse una llamada a la función siguiente para desagrupar una sesión previamente agrupada:

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

Semántica:

Se ejecutará el siguiente código en notación 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;
/* se informa al trayecto de vídeo seguro de la desagrupación de sesiones */

return ErrOk;
```

Precondiciones:

- Las sesiones han sido agrupadas previamente.

La función para iniciar una sesión de encriptación es:

```
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)
```

Semántica:

Se ejecutará el siguiente código en notación C:

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

/* localiza un sessionId libre; cualquier algoritmo es válido */
int i=0;
while (i<NSESSIONS && ss[slotId].se[i].active) i++;
if (i==NSESSIONS) return ErrNoMoreSessions;
/* i contiene un bloque de administración de sesión inactivo */

/* verifica si se ha utilizado una lista de revocación de clientes válida */
if (config.encryptConfig.microServerVersion >
    ss[slotId].clientPOClRLVnr) return ErrRevocEnforce;

*sessionId = i;

/* inicializa la información del estado de la sesión */
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];
}

/* inicializa el estado de clave aleatoria */
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);

/* inicializa el estado importación */
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;
```

Precondiciones:

- El **Intervalo AS** ha sido inicializado satisfactoriamente.

El **Anfitrión ECI** puede hacer pasar la clave aleatoria a un nuevo estado (haciendo que el siguiente pase a ser el actual) de una sesión mediante la función siguiente:

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

Semántica:

Se ejecutará el siguiente código en notación 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;
```

Precondiciones:

- El **Intervalo AS** ha sido inicializado satisfactoriamente.

Cuando funcione en modo TS (flujo de transporte) el trayecto de vídeo seguro señalizará al **Cliente ECI** asociado el cambio de la palabra de control actual por la siguiente (véase [UIT-T J.1012]). El **Cliente ECI** puede utilizar este mensaje para iniciar el cálculo de la siguiente palabra de control.

El **Anfitrión ECI** puede detener la sesión y, en consecuencia, terminar cualquier **Conexión de Exportación** pendiente de esa sesión mediante la siguiente función:

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

Semántica:

Se ejecutará el siguiente código en notación C:

```
int i, j;

ss[slotId].se[sessionId].active = false;

/* desagrupa sesiones de desencriptación previamente agrupadas */
for (j=0; j<NSESSESIONS; j++)
    if (ss[slotId].se[j].coupledSessionId == sessionId)
        ss[slotId].se[j].coupledSessionId = -1;
        /* se informa de la desagrupación al trayecto de vídeo seguro */

/* suprime todas las sesiones de exportación */
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;
```

Precondiciones:

- La sesión del **Intervalo AS** ha sido inicializada satisfactoriamente.

8.2.4.4 Control de exportación del Intervalo AS

El **Mecanismo de Autenticación** de exportación permite al **Anfitrión ECI** crear una **Conexión de Exportación** entre una sesión de **Intervalo AS** en un **Cliente ECI** que realiza una desencriptación y una sesión de **Intervalo AS** de un **Microservidor**, permitiendo la transferencia de contenido entre el **Cliente ECI** que realiza la desencriptación y el **Microservidor**. El **Sistema AS** utiliza el **Subsistema de Procesamiento de Certificados** para procesar la exportación necesaria, las cadenas de autenticación de importación y exportación mediante la POPK de las sesiones del **Intervalo AS** exportador y la minClientVersion como base para validar la exportación y subsiguiente cadena de importación. El resultado final es la validación positiva del elemento de la **Conexión de Exportación** para un ID de grupo de exportación, o bien el rechazo de la conexión. La conexión real se crea entre una sesión (de exportación) y una sesión de importación.

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

Semántica:

expCh es la cadena de exportación desde la POPK al TPEGC o **Certificado** ESC. ImpCh es la cadena de importación desde el TPEGC al ESC.

NOTA – impCh puede estar vacío. auth[] es la secuencia de cadenas de autenticación de exportación necesarias para coautenticar secciones de la cadena de importación.

La estructura CertSerialChain se define en la cláusula 9.5.2.4.2 de [UIT-T J.1012].

El primer **Intervalo AS** verifica impCh utilizando las cadenas de autenticación de exportación auth[] y mediante la **Clave Raíz CEI** instalada y el número de versión de la **Lista de Revocación**.

El **Intervalo AS** solicita entonces al **Subsistema de Procesamiento de Certificados** que procese la cadena de exportación e importación utilizando la POPK, la **POPK** de los registros de estado AS y la **ExportRlVersion** como raíz. El id del primer **Certificado** de la cadena de exportación será almacenado en expGrpId.

Tras una autenticación exitosa, se añade un elemento de exportación al estado de la sesión del **Intervalo AS**, con el id del grupo de exportación y el id del intervalo más el id de sesión del **Cliente ECI** de exportación autenticado. El código en notación C siguiente se ejecutará para crear la conexión de importación. La autenticación puede calcularse para dos id de grupo de exportación, de forma que se permita el paso sin solución de continuidad de un grupo de exportación al siguiente en las propiedades del contenido.

```
/* el CPS establece las siguientes variables tras el procesado exitoso de las cadenas de exportación
– importación */
PubKey impSpk;           /* spk del sistema importador */
uint   impConfigVersion; /* número de versión de config. del sistema de exportación */
uint   expGrpId;         /* grupo de exportación para el que es válida la conexión de exportación */


/* verifica si es aceptable el estado del intervalo de importación potencial */
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;
}

/* verifica si ya existe otra conexión de importación */
if (ss[impSlotId].se[impSessId].ies.importSlotId != ImportNone)
  return ErrExportOngoing;
/* fija el estado de importación/exportación de la sesión de importación para reflejar la conexión
de exportación */
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;
```

Precondiciones:

- La sesión del **Intervalo AS** ha sido inicializada satisfactoriamente.

Una vez establecida una **Conexión de Exportación,** también puede terminarse desde el lado de la importación (que detendrá la sesión de encriptación):

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

Semántica:

Se ejecutará el siguiente código en notación 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;
```

Precondiciones:

- Se ha establecido la sesión del **Intervalo AS** para importación.

8.2.4.5 Inicialización de LK1 de la Escalera de Claves

Para realizar las operaciones del mecanismo de **Escalera de Claves** en un **Intervalo AS**, el **Anfitrión ECI** puede cargar la clave de enlace de nivel superior LK1 para sucesivos cálculos de salida de la **Escalera de Claves**.

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

Semántica:

Se ejecutará el siguiente código en notación C:

```
if (ss[slotId].slotMode == SlotModeEncr) spkIndx = 0;

if (spkIndx >= 16) return ErrParam5;
/* comprueba si spkUri está puesto a 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;
```

Precondiciones:

- Se ha inicializado una sesión de **Intervalo AS**.

8.2.4.6 Cálculo de la palabra de control de encriptación

Una vez fijado el lk1 del campo estado del **Intervalo AS**, pueden calcularse las palabras de control. cwIndx indica la palabra par o impar calculada. El valor puede ser 0 (par) o 1 (impar), siendo siempre 0 para la desencriptación basada en ficheros.

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

Semántica:

Se ejecutará el siguiente código en notación C:

```
PubKey spk[MaxSpkEncr+1], popk[MaxSpkEncr+1]; /* variables temporales */
SessionConfig config[MaxSpkEncr+1]; /* variable temporal*/

/* verificaciones de coherencia básicas */
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;
}

/* verifica si la configuración del intervalo ha sido autenticada */
if (ss[slotId].se[sessId].configAuthMode != ConfigAuthModeAk1)
    return ErrNoConfigAuth;

/* verifica si el estado de la Raíz del Anfitrión ECI del CPS es suficiente para proceder */
if (!cpsEciRootStateOk(slotId,sessId)) return ErrRevocEnforce;


/* verifica si debe aplicarse una clave de sesión aleatoria del intervalo */
SymKey rkAppl; /* clave aleatoria que puede ser necesario aplicar */
if (rkIndx == 0) {
        rkAppl = ss[slotId].se[sessId].rkState.rkCurrent;
} else if (rkIndx == 1) {
        rkAppl = ss[slotId].se[sessId].rkState.rkNext;
} else {
        return ErrParam7;
}

/* si es necesario, se inserta la clave aleatoria del intervalo y la clave aleatoria de la sesión */
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;
}

/* calcula input-C, que se inserta en la escalera de claves */
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]);

/* utiliza ARK con el valor 0 */
uchar ark[16] = (uchar){0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/* define spk, popk y las entradas de config a la escalera de claves; se utilizan spk/popk del
intervalo en la posición 0 y una réplica de la configuración del intervalo */

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;
}

/* define valores de spkUri */
ulong spkUri = (0x1<<(nSpk+1)) - 1; /* todas las SPK pueden utilizarse para decodificar claves */

/* realiza el cálculo de la escalera de claves */
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 se envía al recurso de encriptación junto con cwUri, msField1, cpMask y cwIndx */

return ErrOk;
```

Precondiciones:

- Se ha cargado la LK1de la sesión.
- En caso necesario, la sesión del **Intervalo AS** ha sido autenticada.

8.2.4.7 Cálculo de la palabra de control de desencriptación

Una vez fijado el lk1 del campo estado del **Intervalo AS**, pueden calcularse las palabras de control. cwIndx indica la palabra par o impar calculada. El valor puede ser 0 (par) o 1 (impar), siendo siempre 0 para la desencriptación basada en ficheros.

```
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)
```

Semántica:

Se ejecutará el siguiente código en notación C:

```
/* verificaciones de coherencia básicas */
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 ;

/* si es necesario, verifica si la configuración del intervalo ha sido autenticada */
if ( ss[slotId].se[sessionId].config.decryptConfig.akModeAuth &&
     ss[slotId].se[sessionId].configAuthMode != ConfigAuthModeAk1
   ) return ErrNoConfigAuth;

/* verifica si el estado de la Raíz del Anfitrión ECI es suficiente para proceder */
if (!cpsEciRootStateOk(slotId,sessionId)) return ErrRevocEnforce ;

/* garantiza que se aplican las spk, popk y slotConfig del intervalo adecuadas */
spk[si]= ss[slotId].se[sessionId].spk;
popk[si] = ss[slotId].se[sessionId].popk;

/* solo autentica la configuración de desencriptación del intervalo si es necesario */
if ( ss[slotId].se[sessionId].config.decryptConfig.klModeAuth )
    ssConfig[si].decryptConfig = ss[slotId].ssConfig.decryptConfig;

/* en todos los casos autentica los campos klModeAuth y akModeAuth */
config[si].decryptConfig.klModeAuth=
     ss[slotId].se[sessionId].config.decryptConfig.klModeAuth;
config[si].decryptConfig.akModeAuth =
     ss[slotId].se[sessionId].config.decryptConfig.akModeAuth;

/* verifica si es posible que tenga que aplicarse la clave aleatoria de la sesión del intervalo */
SymKey rpAppl; /* clave aleatoria que puede tener que ser aplicada */
if (rkIndx == 0) {
        rpAppl = ss[slotId].se[sessionId].rkState.rkCurrent;
} else if (rkIndx == 1) {
        rpAppl = ss[slotId].se[sessionId].rkState.rkNext;
} else {
        return ErrParam11;
}

/* inserta la clave aleatoria del intervalo si es necesario */
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;
}

/* calcula input-C, es decir, elk[nElk-1] para la autenticación de las propiedades del contenido */
/* verifica si el bit de control basicUri está puesto a 1 */
if (((elk[nElk-1][0]>>2)&0b1) != 0b1) return ErrBasicUriCtrl;
uchar result1[16];
computeField1Decrypt(elk[nElk-2],result1);
computeInputC(result1, field2, elk[nElk-2]);

/* utiliza ARK con el valor 0 */
uchar ark[16] = (uchar){0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/* realiza el cálculo de la escalera de claves */
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 se transfiere a la sesión de recursos de desencriptación junto con cwUri, result1 y cwIndx y
el valor del asa de medios de los estados de las sesiones */

return ErrOk;
```

Precondiciones:

- Se ha cargado la LK1de la sesión.
- En caso necesario, la sesión del **Intervalo AS** ha sido autenticada.

8.2.4.8 Cálculo de akClient y su aplicación

El **Mecanismo de Autenticación** del **Bloque de Escalera de Claves** permite un cálculo seguro de las claves seguras que usa el **Cliente ECI** mediante el **Mecanismo de Autenticación**:

```
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)
```

Semántica:

Se ejecutará el siguiente código en notación C:

```
/* verificaciones de coherencia básicas */
if (ss[slotId].slotMode==SlotModeEncr) spkIndx = 0;
if (spkIndx >= 16) return ErrParam4;
/* comprueba si spkUri en 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;
}
/* garantiza que se aplican la spk y la popk adecuadas para el intervalo */
popk[spkIndx] = ss[slotId].popk;

/* garantiza que se aplican la ACF y la ARK adecuadas */
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} ;
}

/* ejecuta el mecanismo de autenticación */
ss[slotId].akClient =
      AuthMech(inputV,acf,ark,popk,akCnf,XT,spkUri,nSpk,spkIndx,spk);
return ErrOk;
```

Precondiciones:

- El intervalo ha sido inicializado.

Mediante la función siguiente se garantiza el uso de la clave AK del **Cliente ECI** calculada:

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

Semántica:

Se ejecutará el siguiente código en notación C:

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

Precondiciones:

- El intervalo ha sido inicializado.
- El AkClient del intervalo se ha calculado correctamente.

8.2.4.9 Autenticación de la configuración de la sesión del Intervalo AS

El **Mecanismo de Autenticación** del **Bloque de Escalera de Claves** permite que el **Servidor de Aprovisionamiento** autentique la configuración de la sesión del intervalo. El **Servidor de Aprovisionamiento** puede generar información de autenticación fuera de línea o solicitar que se realice la autenticación en línea fijando AkOnline en ACF. Hay disponibles dos funciones distintas para la autenticación de un intervalo de desencriptación y un intervalo de encriptación.

```
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])
```

Semántica:

Se ejecutará el siguiente código en notación C:

```
/* verificaciones de coherencia básicas */
if (!ss[slotId].se[sessionId].active) return ErrParam2;
if (ss[slotId].slotMode!=SlotModeDecr) return ErrSlotMode;
if (spkIndx >= 16) return ErrParam5;
/* comprueba si spkUri en 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;

/* verifica si el estado de la Raíz del Anfitrión ECI del CPS es suficiente para proceder */
if (!cpsEciRootStateOk(slotId)) return ErrRevocEnforce;

/* garantiza que se aplican las spk, popk y config del intervalo adecuadas */
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};
}

/* ejecuta el mecanismo de autenticación */
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;
}
```

Precondiciones:

- Se ha cargado la LK1de la sesión de **Intervalo AS**.

La autenticación a efectos de encriptación incluye verificar el estado específico de la encriptación.

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

Semántica:

Se ejecutará el siguiente código en notación C:

```
PubKey spk[MaxSpkEncr+1], popk[MaxSpkEncr+1]; /* variables temporales */
SessionConfig config[MaxSpkEncr+1]; /* variable temporal */

/* verificaciones de coherencia básicas */
if ((ss[slotId.SlotMode != SlotModeEncr) return ErrSlotMode;

/* verifica si el estado de Raíz del Anfitrión ECI del CPS es suficiente para proceder */
if (!cpsEciRootStateOk(slotId,sessId)) return ErrRevocEnforce;

/* define las entradas spk, popk y config a la escalera de claves; se utilizan spk/popk del
intervalo en la posición 0 y una réplica de la configuración del intervalo */

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;
}

/* define valores de spkUri */
ulong spkUri = (0x1<<(nSpk)) - 1;
                      /* todas las SPK pueden utilizarse para la decodificación del contenido */

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};
}

/* ejecuta el mecanismo de autenticación */
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;
}
```

Precondiciones:

- Se ha cargado la LK1de la sesión del **Intervalo AS**.

8.2.4.10 Carga de la clave secreta del Microservidor

Un cliente **Microservidor** puede utilizar el **Intervalo AS** en modo servidor asimétrico y cargar un valor de clave secreta ussk del **Microservidor** para el subsiguiente establecimiento de una conexión segura con un conjunto de circuitos integrados (chipset) del **Microcliente**:

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

Semántica:

Se ejecutará el siguiente código en notación C:

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

/* verificaciones de coherencia básicas */
if (ss[slotId].slotMode!=SlotModeEncr) return ErrSlotMode;
if (!ss[slotId].se[sessId].config.encryptConfig.asymKlMode)
        return ErrSlotModeUndefined;

/* verifica si el estado de Raíz del Anfitrión ECI del CPS es suficiente para proceder */
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;
}
/* define valores de spkUri */
ulong spkUri = (0x1<<(nSpk+1)) - 1; /* todas las SPK pueden utilizarse para decodificar claves */

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};
}

/* ejecuta el mecanismo de autenticación */
Secret SymKey ak =
      AuthMech(inputV,acf,ark,popk,config,XT,spkUri,nSpk,0,spk);

/* realiza la decodificación AES ECB de ussk */
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;
```

Precondiciones:

- Se ha autenticado la configuración de la sesión.

8.2.4.11 Generación de MinitLk1 para Microclientes

En el modo **Microservidor** asimétrico el **Intervalo AS** puede generar mensajes de inicialización del **Bloque de Escalera de Claves** para **Microclientes**:

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

Semántica:

ECI_Certificate_Chain se define en la cláusula 5.4.1 de [UIT-T J.1012], y contiene la **Cadena de Certificados** para validar un **Microcliente**. Esta función utiliza en primer lugar el CPS para validar ClCPK utilizando slot[slotId] con POPK como **Certificado Padre** y ss[SlotId].se[sessId]. config.encryptConfig.microServerVersion como número mínimo de versión de la **Lista de Revocación** para el primer **Certificado** de la cadena. Si esta validación tiene éxito la clcpk variable contiene la clave pública del conjunto de circuitos integrados del cliente, y se ejecuta el siguiente código en notación C:

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

Precondiciones:

- Ussk es inicializada.
- La sesión está en modo de encriptación asimétrica.

8.2.4.12 Cálculo de la clave de desencriptación de la imagen de Cliente ECI

A fin de realizar la carga de una imagen encriptada, el **Intervalo AS** puede proporcionar una clave de autenticación que permite desencriptar la clave para la decodificación. Esta función debe realizarse antes de la inicialización del intervalo:

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

Semántica:

Se ejecutará el siguiente código en notación C:

```
/* se utiliza un estado de configuración del intervalo por defecto */
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;

/* crea directamente popk/spk, XT, clCnf, */
PubKey popkArr[1]; /* también utilizado por 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} ;
}

/* ejecuta el mecanismo de autenticación */
```

```
Secret SymKey ak =
      AuthMech(inputV,acf,ark,popkArr,cnf,XT,spkUri,1,0,popkArr);

Secret SymKey dImgKey = AuthMechResponse(ak, eImgKey);
/* el cargador del cliente utiliza dImgKey para desencriptar la imagen de cliente utilizando el modo
AES CBC con IV=0 */

return ErrOk;
```

Precondiciones:

- El intervalo se fija en el estado por defecto; slotRk se fija en un nuevo valor aleatorio.

NOTA – Esta función no se ejecuta a petición del **Cliente ECI**.

8.2.4.13 Lectura de la información de Seguridad Avanzada

El **Sistema AS** proporciona al **Cliente ECI** el acceso a los datos que genera y una clave aleatoria de propósito general.

NOTA 1 – Las funciones "get" (obtener) y "set" (fijar) definidas en esta cláusula no generan errores automáticos para valores indefinidos de parámetros; en el caso de las funciones "get" simplemente se devuelve un valor indefinido y en el caso de las funciones "set" no tiene efecto alguno.

La función siguiente lee la clave aleatoria del **Intervalo AS** (normalmente utilizada como palabra ocasional (nonce) para las sesiones):

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

Semántica:

Se ejecuta el siguiente código en notación C:

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

Si el intervalo no se inicializa devuelve un número.

La función siguiente lee el estado de la clave aleatoria de la sesión:

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

Semántica:

Se ejecuta el siguiente código en notación C:

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

Si el intervalo no se inicializa devuelve un número.

Puede leerse el contador de límite de la clave de sesión aleatoria:

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

Semántica:

Se ejecuta el siguiente código en notación C:

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

Si el intervalo no se inicializa devuelve un número.

Puede fijarse un valor del contador de límite en base al cual se genera un evento (por ejemplo, la renovación de la clave aleatoria con tiempo suficiente):

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

Semántica:

Se genera un evento eventSessionLimitCounter una vez que la condición siguiente pasa a ser verdadera tras una llamada a esta función:

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

NOTA 2 – Una segunda llamada invalida la anterior. La segunda llamada a esta función con un valor muy grande de eventLimit anula el evento (excepto cuando el evento ya hubiera sido generado).

El evento siguiente se genera cuando se alcanza un límite de evento para una sesión:

```
reqAsEventSessionLimit(uint slotId, uint sessionId)
```

NOTA 3 – Este evento se traduce en un mensaje síncrono que carece de la correspondiente contestación en [UIT-T J.1012].

8.2.4.14 Generación de números aleatorios de cliente

El **Cliente ECI** puede solicitar un número aleatorio de 128 bits generado por el **Sistema AS** mediante una llamada a la siguiente función:

```
SymKey getAsClientRnd()
```

Semántica:

Se ejecuta el siguiente código en notación C:

```
return rnd128();
```

8.2.4.15 Códigos de error

Los valores de los códigos de error que devuelve la función descrita en la cláusula 8.2.4 se definen en el Cuadro 8-14.

Estos códigos de error respetan el convenio de códigos de error de los mensajes entre un **Anfitrión ECI** y un **Cliente ECI** definido en la cláusula 9 de [UIT-T J.1012].

Cuadro 8-14 – Definición de códigos de devolución de errores

Código de error de retorno	Valor	Descripción
ErrSlotMode	–256	El **Intervalo AS** no está en un modo adecuado para esta operación.
ErrNoMoreSessions	–257	No hay más sesiones disponibles.
ErrSession1Coupled	–258	La primera sesión ya ha sido agrupada.
ErrSession2Coupled	–259	La segunda sesión ya ha sido agrupada.
ErrSessionNotCoupled	–260	Sesión no agrupada.
ErrNoSuchSession	–261	Sesión inexistente.
ErrExportNoSlot	–262	Intervalo de exportación desconocido.
ErrExportSlotBadState	–263	Intervalo de exportación en estado inadecuado.
ErrExportOngoing	–264	Sesión de exportación ya tiene una **Conexión de Exportación**.
ErrImportSlotBadState	–265	El intervalo de encriptación no está en modo encriptación.
ErrNoExport	–266	No hay una exportación en curso para la sesión.
ErrSpkUriViolation	–267	SpkUri para la SPK del intervalo tiene un valor inadecuado para el modo del intervalo.
ErrSlotModeUndefined	–268	El modo del intervalo tiene un valor inadecuado para esta operación.
ErrRevocEnforce	–269	La **Revocación ECI** no permite el funcionamiento del intervalo.
ErrNoConfigAuth	–270	La configuración del intervalo no ha sido autenticada adecuadamente.
ErrNoSlotRkInsert	–271	El vector ELK no es lo suficientemente extenso para insertar una clave aleatoria.
ErrSpk0NoDecrypt	–272	spk[0] no puede usarse para generar palabras de control de desencriptación.
ErrBasicUriCtrl	–273	No se ha fijado el valor del bit de control de field1 de la URI básica.
ErrOk	0	Llamada exitosa.
ErrSlotConfigAuthFail	–274	Ha fallado la autenticación de la configuración de sesión del intervalo.
ErrParam<N>	**–<N>**	Error en el parámetro de entrada **N** (ErrParam1 tiene el valor –1 y señaliza un error en el parámetro 1).
	1..MaxInt	Llamada exitosa, el valor viene definido por las definiciones del mensaje.

9 Aleatorización/desaleatorización y exportación de contenido

9.1 Funcionalidad básica

El **Trayecto de Vídeo Seguro** puede desencriptar el contenido. Este contenido está acompañado de **Propiedades del Contenido** y de la **Conexión de Exportación**. El contenido puede enviarse a salidas normalizadas si las **Propiedades del Contenido** lo permiten, y pueden ser reencriptadas por un **Microservidor** en caso de que exista correspondencia con la **Conexión de Exportación**.

Con fines de gestión de recursos, la **ECI** define recursos de desencriptación y de encriptación. Un recurso se utiliza para desencriptar o encriptar contenido de una única sesión de medios encriptada o a encriptar con una única CW en cada momento; un recurso de encriptación o desencriptación permanece conectado a un único **Intervalo AS** de desencriptación o encriptación. En el caso de desencriptación de flujo TS, el recurso de desencriptación tiene una memoria intermedia dual para una CW par e impar. La CW par o impar es seleccionada por el flujo a decodificar. Ello puede exigir el cambio de la palabra de control sobre la marcha si cambian las **Propiedades del Contenido** del contenido a encriptar. Para desencriptación o encriptación de ficheros el **Anfitrión ECI** proporciona la sincronización entre la CW y el contenido a desencriptar, algo que puede ser sustancialmente más rápido que el tiempo real. Los recursos de desencriptación y reencriptación de ficheros sólo requiere una única memoria intermedia para la CW.

NOTA – Los flujos TS que requieren dos o más palabras de control para desaleatorizar distintos flujos elementales necesitan múltiples recursos de desaleatorización y, por lo tanto, varias sesiones.

La **ECI** no especifica condiciones específicas relativas a la memoria intermedia o al (potencialmente amplio) procesamiento intermedio, como la transcodificación o las marcas de agua que pueden aplicarse al contenido desencriptado que pasa del recurso de desencriptación al de encriptación. Esos procesos pueden causar demoras importantes. Los fabricantes de CPE pueden seleccionar las implementaciones adecuadas que causan un desplazamiento temporal entre un recurso de desencriptación y un recurso de reencriptación conectado. El intervalo de reencriptación y el **Cliente ECI** se sincronizan con la encriptación del contenido.

9.2 Especificaciones del aleatorizador y el desaleatorizador

La función de desaleatorización de un CPE **ECI** soportará los siguientes algoritmos de desaleatorización en modo TS:

- CSA1/2, en los modos PES y TS tal como se define en [ETSI ETR 289] y [b-ETSI DVB CSA].
- CSA3, en los modos PES y TS [ETSI TS 100 289] y [b-ETSI DVB CSA3].
- DVB-CISSA en los modos PES y TS [ETSI TS 103 127].

La función de desaleatorización de un CPE **ECI** soportará los siguientes algoritmos de desaleatorización en modo fichero:

- Modo CENC AES128 CTR y modo AES128-CBC (en ambos casos encriptación completa de muestra y de submuestra), tal como se define en [ISO/CEI 23001-7]. CENC y [ISO/CEI 23009-4] para MPEG-DASH.

La función de aleatorización de un CPE **ECI** soportará los siguientes algoritmos de desaleatorización en modo TS:

- DVB-CISSA en modos PES y TS [ETSI TS 103 127].

La función de aleatorización de un CPE **ECI** soportará los siguientes algoritmos de desaleatorización en modo fichero:

- Modo CENC AES128 CTR y modo CBC (en ambos casos encriptación completa de muestra y de submuestra), tal como se define en [ISO/CEI 23001-7] y [ISO/CEI 23009-4]. El trayecto de vídeo seguro generará un único vector de inicialización para contenido encriptado con una única CW en el modo AES-CTR y seguirá las reglas de definición IV para el modo AES-CBC, tal como se define en [ISO/CEI 23009-4]. El **Anfitrión ECI** puede acceder a los vectores de inicialización a fin de utilizarlos para empaquetar contenido.

9.3 Control de la exportación

Las sesiones del **Intervalo AS** de desencriptación utilizan **Conexiones de Exportación** autenticadas a modo de boletos para autorizar la importación y exportación mediante el recurso de desencriptación. Un recurso de desencriptación permitirá la exportación de contenido desencriptado a un recurso de encriptación si la **Conexión de Exportación** que proporciona la sesión del **Intervalo AS** asociado lo permite para un ID de grupo de exportación, tal como se define en la cláusula 8.2.4.4. Un recurso de desencriptación no permitirá la exportación de contenido desencriptado a un recurso de encriptación si la **Conexión de Exportación** del grupo de exportación seleccionado según los ID del grupo de exportación de las **Propiedades del Contenido** no es una **Conexión de Exportación** validad proporcionada por el **Intervalo AS** asociado.

9.4 Control de salida

Las **Propiedades del Contenido** del control de salida se utilizan para deshabilitar o habilitar la exportación de contenido en el marco de la protección que ofrecen las tecnologías de protección normalizadas de la industria en las conexiones de salida del CPE. Un recurso de desencriptación permitirá la exportación de contenido desencriptado a una salida si lo permite la información de control de la salida de la sesión del **Intervalo AS** asociado. Un recurso de desencriptación no permitirá la exportación de contenido desencriptado a una salida si la sesión de **Intervalo AS** asociada no incluye el permiso en la información de control de salida.

9.5 Comparación de las Propiedades del Contenido de sesiones agrupadas

El **Trayecto de Vídeo Seguro** verificará que las **Propiedades del Contenido** definidas en el field1 de una sesión, excluyendo los dos primeros bytes, son iguales a las **Propiedades del Contenido** de cualquier sesión agrupada. Se permitirá la exportación y salida de contenido de sesiones agrupadas con las mismas **Propiedades del Contenido**. A partir de ese momento, los flujos combinados se tratarán como una sesión desde perspectiva de la protección **ECI**. La agrupación de flujos estará inhabilitada si no son iguales las **Propiedades del Contenido** de field1, excluyendo los dos primeros bytes.

9.6 Propagación de las Propiedades del Contenido en la exportación

La sesión de recursos de desencriptación propagará las **Propiedades del Contenido** de field1 fijadas por el cliente y (parcialmente) autenticadas implícitamente por la **Escalera de Claves** junto con el contenido de los recursos de la sesión de reencriptación que importan el contenido desencriptado, tal como se define en la cláusula 8.2.4.6. Las sesiones de encriptación que reciben el contenido desencriptado comparan los bytes del field1 designado con el valor de field1 para la encriptación del contenido, al tiempo que se aplica una máscara para seleccionar los campos que deben ser propagados con arreglo a la función, asegurando por lo tanto que los bytes field1 del cliente de desencriptación designado se propaguen al contenido encriptado.

El siguiente código en notación C será ejecutado por la sesión de recursos de exportación con cada cambio en los valores de entrada impField1, expField1 y cpMask:

```
uchar impField1[16];  /* valores de field1 para el contenido importado */
uchar expField1[16];  /* valores de field1 del cálculo de la CW de encriptación */
ushort cpMask;        /* mascara de la comparación */

bool propOk = true;  /* indica si la propagación de contenido importado es correcta */
int i;

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

if (propOk) /* contenido de reencriptación */
else /* no se reencripta el contenido */
```

9.7 Aplicación de la URI básica en la exportación

La propagación de la URI básica desde la sesión del **Intervalo AS** de desencriptación a la sesión del **Intervalo AS** de reencriptación se controla mediante los mecanismos siguientes, siendo slotId el ID del intervalo de encriptación y sessionId el ID de la sesión de encriptación.

Los derechos asignados al contenido para la URI básica por el **Intervalo AS** de encriptación no presentan mayor flexibilidad que los asociados al contenido propagado.

El **Microservidor** es autenticado: ss[slotId].se[sessionId].config.decryptConfig,akModeAuth es igual a 0b1.

Si la URI básica no permite la reproducción del contenido (es decir, modo flujo) durante la exportación se verifica lo siguiente:

– ss[slotId].se[sessionId].config.decryptConfig.rkDecrMode.mode será distinto de RKModeNone (es decir, se aplica un dato aleatorio de un solo uso ("nonce") para impedir la reproducción de contenido previamente codificado en el rearranque de un sistema); y

– ss[slotId].se[sessionId].klModeAuth se fijará a 1 (valor 0b1) garantizando que la decryptConfig utilizada por el servidor, incluida la inserción de una clave aleatoria en el **Microcliente**, se autentica al tiempo que es utilizada por el **Microcliente** sobre la base del cálculo de la **Escalera de Claves**.

9.8 Aplicación de las Propiedades del Contenido en salidas normalizadas de la industria

Una salida normalizada, normalmente compuesta por una salida física combinada con un sistema de protección estándar de la industria, utilizará las **Propiedades del Contenido** para seleccionar la protección adecuada de la salida o para deshabilitar la salida si no es posible fijar valores adecuados. Las reglas precisas al respecto se definen en las reglas de conformidad.

La **Robustez** de la implementación de la URI básica y de las **Propiedades del Contenido** del control de salida será de un nivel similar a la del trayecto de vídeo seguro.

La **Robustez** de la aplicación de la URI estándar será al menos tan elevada como la de la implantación del **Anfitrión ECI**, con la excepción de las funciones con requisitos de implementación complejos.

9.9 Sincronización de la palabra de control

Para procesar flujos TS el **Trayecto de Vídeo Seguro** proporciona las funciones siguientes que permiten controlar los cambios de las palabras de control (para la encriptación) y proporcionar notificaciones relativas a los cambios del campo control de aleatorización. Las funciones y eventos de esta sesión cumplen los convenios definidos en la cláusula 8.2.4.

La sesión del **Intervalo AS** puede proporcionar una palabra de control "par" y una "impar" para su aplicación a la encriptación o desencriptación del contenido.

En caso de desencriptación, el campo control de aleatorización [ETSI TS 100 289] informa a la función de desencriptación sobre la palabra de control que debe utilizar. Si el contenido se señaliza como no aleatorizado, no se utilizará ninguna palabra de control. El valor del resultado es igual al campo control de aleatorización, con valores definidos en la cláusula 5.1 de [ETSI TS 100 289].

La función siguiente lee el estado actual del campo control de aleatorización en el flujo:

```
uint getAsSC(uint slotId, uint sessionId)
```

En caso de encriptación la palabra de control aplicada puede cambiar por la ocurrencia de dos eventos:

1) Un cambio en las **Propiedades del Contenido** del contenido importado, que provocará un cambio en la palabra de control aplicada para la encriptación. Este cambio puede ser demorado por el **Intervalo AS** a fin de completar un cambio en curso de la palabra de control causado por el evento siguiente.

   Una señal de la sesión de **Intervalo AS** que la palabra de control aplicada tiene que modificar.

Si el contenido importado no está aleatorizado, no se aplicará aleatorización alguna para la encriptación, fijándose el valor del campo control de aleatorización de contenido en 0b00 en la primera ubicación de cambio posible. A la viceversa, si el contenido importado pasa de no aleatorizado a aleatorizado, el contenido se aleatorizará con la siguiente palabra de control; se seleccionará la clave opuesta con relación al contenido que estaba aleatorizado antes de la sección no cifrada del contenido.

El evento que señala el cambio de la propiedad del contenido importado se define como:

```
reqAsEventCpChange(uint slotId, uint sessionId)
```

Semántica:

El evento señaliza un cambio en las **Propiedades del Contenido** del contenido importado si dicho contenido requiere encriptación.

El **Trayecto de Vídeo Seguro** no permitirá discrepancia alguna entre los parámetros de la encriptación y las **Propiedades del Contenido** importadas para un periodo de tiempo mayor. En la cláusula 6.6.2 de [b-UIT-T J. Supl. 7] se propone un valor máximo.

NOTA 1 – Este evento no se producirá si el contenido importado pasa de encriptado a no encriptado. Las **Propiedades del Contenido** no se aplican a un contenido no encriptado.

El **Trayecto de Vídeo Seguro** permite al **Sistema AS** evitar que se produzca el cambio automático de un eventCpChange de las **Propiedades del Contenido** mediante la instrucción siguiente:

```
setAsPermitCPChange(uint slotId, uint sessionId, bool permit)
```

Semántica:

Esta función fija el valor del permiso al objeto de permitir un cambio automático de las propiedades de control del contenido importado para provocar un cambio en la palabra de control del contenido encriptado.

NOTA 2 – Esta función debería preceder a cualquier palabra de control posterior no calculada sobre la base de las **Propiedades del Contenido** que siguen, por ejemplo, reflejando únicamente un cambio de palabra aleatoria de un solo uso ("nonce") o de clave aleatoria.

NOTA 3 – Si el permiso para realizar cambios está inhabilitado (`permit==falso`), deberá restaurarse durante el tiempo permitido para el caso en que se produzca una discrepancia con las **Propiedades del Contenido** del contenido importado al objeto de no interrumpir el flujo reencriptado.

La siguiente función permite poner el campo control de aleatorización de la encriptación en un determinado estado:

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

Semántica:

El campo control de aleatorización se fija en el valor de scramblingControlField del primer punto posible de cambio en el flujo. Sólo se permiten que scramblingControlField tome los valores 0b10 y 0b11 (aleatorización con clave par e impar respectivamente).

El campo control de aleatorización del flujo encriptado se fijará en 0b00 (sin aleatorización) si el estado del contenido importado es no encriptado.

Para las sesiones de desencriptación y encriptación se define la función de evento siguiente:

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

Semántica:

Este evento se produce cuando el campo control de aleatorización cambio de estado.

10 Subsistema de Procesamiento de Certificados

10.1 Reglas básicas de procesamiento de cadenas de certificados

El **Subsistema de Procesamiento de Certificados** puede procesar **Cadenas de Certificados** para autenticar elementos sobre la base de una clave pública inicial y un número mínimo de la **Lista de Revocación**. La mayor parte del procesamiento de una cadena de certificados es genérico. En esta cláusula se definen las reglas genéricas de procesamiento para **Cadenas de Certificados**. En las cláusulas siguientes se definen reglas específicas de procesamiento para varios tipos de cadenas.

Las **Cadenas de Certificados** siguientes se definen en la cláusula 5.4 de [UIT-T J.1012].

La definición de reglas de CPS utiliza un enfoque paso a paso para procesar las **Cadenas de Certificados**, comenzando con el inicio de la cadena (la primera **Lista de Revocación**) utilizando una clave pública inicial y el número mínimo de la **Lista de Revocación**. El primer paso es verificar la **Lista de Revocación.** El segundo paso verifica el siguiente **Certificado** de la cadena. Una vez realizados los pasos 1 y 2, se definen una nueva clave pública y un nuevo número de **Lista de Revocación** para procesar el resto de la cadena. Los pasos 1 y 2 se repiten hasta que se ha procesado toda la cadena. Por lo general, se recomienda que las funciones software que ofrecen cadenas las validen previamente a fin de evitar un fallo imprevisto del CPS durante el procesamiento de una cadena.

Los pasos del procesamiento genérico de una **Cadena de Certificados** son los siguientes:

1) El CPS realizará la siguiente verificación de la Lista de Revocación:
    a) El CPS verificará el campo **format_version** de la **Lista de Revocación** para cotejar que es una versión que puede interpretar (véanse las reglas específicas de procesamiento de cadenas) y los campos **rl_id.type** y **rl_id.rl_indicator** para cotejar que se trata de valores esperados.
    b) Si el **Padre** es un **Certificado Raíz** (**root_version_indicator**=1) el **Anfitrión ECI** seleccionará el **Certificado Raíz** cuya **root_version** sea el **Padre**, en caso contrario, se utiliza el **Certificado** precargado o precedente.
    c) El CPS verificará la firma de la **Lista de Revocación** con la última clave pública validada.
    d) El CPS verificará si la longitud de la **Lista de Revocación** se corresponde con sus valores de campo y que todos los campos longitud variable tienen una longitud adecuada.
    e) El CPS verificará si el número de versión de la **Lista de Revocación** no ha sido invalidado por el número mínimo de la **Lista de Revocación**.

2) El CPS realizará la siguiente verificación del Certificado:

   a) El CPS verificará si ***next*** <type, entity_ id, version> del **Certificado** de la cadena no ha sido revocado con arreglo a la **Lista de Revocación** y establece la versión mínima de la **Lista de Revocación** que debe acompañar a ese **Certificado** según los campos **base_rl_version** y **min_rl_version** de la última **Lista de Revocación**.

   b) El CPS verificará el campo **format_version** de la **Lista de Revocación** para cotejar que se trata de una versión que puede interpretar.

   c) El CPS verificará si la longitud del certificado corresponde con sus valores de campo y que todos los campos longitud variable tienen una longitud adecuada.

   d) El CPS verificará la firma del **Certificado** con la clave pública.

Después de los pasos de procesamiento 1) y 2) se actualizan la clave pública y el valor mínimo de la **Lista de Revocación.** La clave pública será igual al campo clave pública del **Certificado** procesado en el paso 2 y el valor mínimo de versión de la **Lista de Revocación será el** identificado en el paso 2a.

No es necesario que todos los **Certificados** están acompañados de una lista de revocación. Si el bit más significativo del campo tipo (type) de un id de certificado es cero, el **Subsistema de Procesamiento de Certificados** requerirá que una **Lista de Revocación** acompañe al **Certificado** para el ulterior procesamiento de la cadena. En los pasos anteriores no se aplicará ningún procesamiento de la **Lista de Revocación** y del número de versión así como de los números de versión de la **Lista de Revocación**, si no es necesaria ninguna **Lista de Revocación**.

10.2 Reglas específicas para cadenas de imágenes de anfitrión

En la validación específica de cadenas de imagen de anfitrión el CPS verifica que:

1) La primera **Lista de Revocación** sea del tipo 0x1 (Lista de Revocación de Fabricantes).
2) El primer **Certificado** sea del tipo 0x1 (**Certificado** de Fabricante).
3) La segunda **Lista de Revocación** sea del tipo 0x0 (**Lista de Revocación** del **Anfitrión ECI**).
4) El segundo **Certificado** sea del tipo 0x0 (**Certificado** de **Anfitrión ECI**).
5) Un posible tercer **Certificado** sea del tipo 0x98 (**Certificado** de series de imágenes de **Anfitrión ECI**).

La clave pública del último **Certificado** (ya sea **Certificado** de **Anfitrión ECI** o **Certificado** de series de imágenes de **Anfitrión ECI**) se utilizará para validar la imagen real del **Anfitrión ECI**.

10.3 Reglas específicas para cadenas de imágenes de cliente

En la validación específica de cadenas de imagen de cliente el CPS verifica que:

1) La primera **Lista de Revocación** sea del tipo 0x2 (suministrador de **Lista de Revocación**).
2) El primer **Certificado** sea del tipo 0x2 (**Certificado** de suministrador).
3) La segunda **Lista de Revocación** sea del tipo 0x0 (**Lista de Revocación** de **Cliente ECI)**.
4) Un posible segundo **Certificado** sea del tipo 0x1 (**Certificado** de series de Cliente).

La clave pública del último **Certificado** (ya sea **Certificado** de suministrador o **Certificado** de series de imágenes de Cliente) se utilizará para validar la imagen real del **Anfitrión ECI**, teniendo en cuenta el último número de versión de la **Lista de Revocación** de Cliente para verificar la versión de la imagen en caso de que el último **Certificado** sea el **Certificado** del suministrador.

10.4 Reglas específicas para certificados de Operación de Plataforma

En la validación específica de **Cadenas de Certificados de Operación de Plataforma** el CPS verifica que:

1) La primera **Lista de Revocación** sea del tipo 0x3 (Lista de Revocación de **Operador**).
2) El primer **Certificado** sea del tipo 0x3 (**Certificado** de **Operador**).
3) La segunda **Lista de Revocación** sea del tipo 0x0 (**Lista de Revocación** de **Operación de Plataforma**).
4) El segundo **Certificado** sea del tipo 0x0 (**Certificado** de **Operación de Plataforma**).

10.5 Reglas específicas para cadenas de exportación/importación

10.5.1 Procesamiento de cadenas de autorización de exportación

Se suministrará al CPS la cadena de autenticación de exportación y la correspondiente sección de la cadena de exportación de un tercero.

El CPS comenzará con la versión de raíz mínima y la versión de **Lista de Revocación** definida en ss[slotId].se[sessionId].config.decryptConfig.minEciRootState. Procesará la cadena de **Certificados** EAOC y EAC y las **Listas de Revocación** asociadas para verificar el cumplimiento de las reglas específicas siguientes para esa cadena:

- El id de la RL raíz es 0x4 (Lista de Revocación de **Operador** de Autorización de Exportación).
- El id del siguiente **Certificado** (EAOC) es 0x4.
- El id de la siguiente **Lista de Revocación** (REAOC RL) es 0x0.
- El id de los siguientes **Certificados** (EAC) de la cadena es 0x0.
- El contenido del campo extensión del **Certificado** será igual al correspondiente **Certificado** de cadena de exportación de la cadena de exportación.
- El id de la siguiente **Lista de Revocación** (EAC-RL) de la cadena es 0x0.
- Todos los **Certificados** de la cadena de exportación serán validados secuencialmente por la cadena de Autorización de Exportación.
- El primer **Certificado** de una sección de cadena de exportación de un tercero será un TPEGC (id de certificado 0x5).
- El último **Certificado** de una sección de cadena de exportación de un tercero será un TPEGC, ESC o ERC (id de certificado 0x5, 0xE y 0xF respectivamente).
- Los **Certificados** intermedios serán todos EGC (id de certificado 0x4).
- Si el último **Certificado** es un TPEGC, significa que es el inicio de una sección siguiente de cadena de exportación. El proceso de verificación anterior se repetirá para todas las secciones de cadenas de autenticación de exportación y secciones de cadenas de exportación de terceros subsiguientes hasta obtener una sección de cadena de exportación de tercero validada (que finaliza con un ESC o un ERC).

10.5.2 Verificación de cadenas de exportación

El CPS comenzará utilizando la clave pública POC, el índice de grupo de exportación para el que se establece la exportación y el número mínimo de versión de la **Lista de Revocación** que debería aplicarse a la **Lista de Revocación de** POC tal como figura en el campo de estado ss[slotId].se[sessionId]. config.decryptConfig.minClientVersion.

NOTA – Esa validación se basa en una autenticación adecuada de la POPK y en la versión de la **Lista de Revocación**. Para ello debe utilizarse la autenticación en modo AK o la autenticación implícita utilizando la **Escalera de Claves** (véase la cláusula 8.2.2.2, campos klModeAuth y akModeAuth).

El CPS procesará el POC-RL, EGC y EGC-RL y el subsiguiente TPEGC o ESC como una **Cadena de Certificados** ordinaria. Se verificarán las reglas adicionales siguientes:

- El tipo de EGC es 0x4.
- El campo export_group_id del EGC será igual al índice de grupo de exportación.
- El tipo de la **Lista de Revocación** EGC es 0x4.
- El tipo de EGC-RL es 0x4.
- El tipo de TPEGC o ESC corresponde al valor del Cuadro 5.2-2 de [UIT-T J.1012].

El procesamiento de un TPEGC se especifica en la cláusula 10.5.3. El procesamiento de un ESC se especifica en la cláusula 10.5.4.

10.5.3 Verificación de cadenas de exportación de terceros

El procesamiento de la cadena de exportación de un tercero comienza con la validación del TPRGC inicial y del número mínimo de versión de la **Lista de Revocación** para su **Lista de Revocación**. El procesamiento terminará con un ESC.

10.5.4 Procesamiento del certificado del sistema de exportación

Para validar la **Conexión de Exportación** se utilizan la SPK del **Certificado** ESC (clave pública del ESC) y el número mínimo de versión de la **Lista de Revocación** del **Padre** del ESC**.** La SPK del **Certificado** deberá concordar con el campo ss[slotId].spk del intervalo de exportación designado. El número de versión mínimo de la **Lista de Revocación** será mayor que el ss[slotId].ssConfig.microServerVersion de exportación.

NOTA – Para garantizar una autenticación correcta, la SPK del intervalo de exportación y la microServerVersion deben ser autenticadas utilizando el **Mecanismo de Autenticación de** AK del **Intervalo AS**.

10.5.5 Reglas de procesamiento de la cadena de un cliente objetivo

El procesamiento de la Cadena de Cliente **Objetivo** comienza con la POPK y la **Lista de Revocación** mínima de un estado **MSConfig** de **Microservidor**. El procesamiento de Cadenas de Cliente **Objetivo** que realiza el CPS seguirá las reglas genéricas definidas en la cláusula 10.1. Además, el procesamiento de la Cadena de Cliente **Objetivo** por el CPS seguirá las reglas específicas siguientes:

1) La primera Lista de Revocación es del tipo 0x0 (Lista de Revocación Objetivo).
2) El primer **Certificado** es del tipo 0x0 (**Certificado** de Grupo **Objetivo**) o 0x8 (**Certificado** de **Microcliente**).
3) Los pasos 1 y 2 se repiten si el certificado del paso 2 es un **Certificado** de Grupo **Objetivo**.

La clave pública del Certificado de **Microcliente** resultante es la clave pública del conjunto de circuitos integrados (chipset) que se utilizará con arreglo al mecanismo descrito en la cláusula 7.3.

10.6 Inicialización de la Clave Raíz ECI del CPS

En el momento de la inicialización del **Sistema AS**, el **Anfitrión ECI** carga en el CPS la información más reciente sobre la **Clave Raíz ECI** y el número de la **Lista de Revocación** aplicables.

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

Semántica:

Se ejecutará el siguiente código en notación C:

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

El CPS aplicará **rootKeyVersion** como el número de versión de la **Clave Raíz ECI** y aplicará el **minRevListNr** a todas las cadenas que se le proporcionan para la carga de credenciales **ECI**.

Todos los demás estados del **Sistema AS** serán reinicializados.

Obsérvese que los valores asignados a ambos parámetros por el **Anfitrión ECI** deben garantizar que todos los **Clientes ECI** pueden cargarse y que el **Anfitrión ECI** no sea revocado, y que ninguno de los **Clientes ECI** sufrirá una **Revocación**.

11 Elementos fundamentales del cargador

11.1 Introducción

El sistema **ECI** utiliza un mecanismo cargador que permite a los **Clientes ECI** verificar de forma segura la versión del **Anfitrión ECI** y las credenciales de **Cliente ECI** que han sido cargadas a fin detectar problemas de seguridad. Ello permite que la actualización del **Anfitrión ECI** y los **Clientes ECI** (tanto las imágenes como las POPK) sea una función ordinaria del sistema.

El cargador de imágenes del **Anfitrión ECI** y del **Cliente ECI** se basa en determinados principios de **Robustez** que se definen en forma de reglas en las cláusulas siguientes. La **Robustez** de la implementación de esas reglas se definirá mediante un documento que queda fuera del alcance de la especificación **ECI**, pero por lo general todas las reglas presentan una **Robustez** de implementación similar. No obstante, se considera que la implementación de algunas reglas tiene una mayor **Robustez** (superior) y que serán son sustancialmente más robustas que la implementación del **Anfitrión ECI**.

11.2 Reglas del cargador de anfitrión

El **Cargador de Anfitrión ECI** cumplirá las reglas siguientes:

1) El **Cargador de Anfitrión ECI** garantizará que la versión de Raíz **ECI** y el número de versión de la **Lista de Revocación** Raíz **ECI** utilizadas para validar las Imágenes del **Anfitrión ECI** se almacenan durante la inicialización que se produce en el encendido del equipo, y que no será posible modificar el número a partir de ese momento. Esta regla requiere una **Robustez** superior.
2) No se podrá modificar el **Cargador de Anfitrión ECI**. Esta regla requiere una **Robustez** superior.
3) No se podrá modificar ni visualizar una imagen de **Anfitrión ECI** una vez que ha sido cargada en tanto que sea necesario para impedir la manipulación de información sensible o la visibilidad de información secreta.
4) La verificación de cualquier imagen de **Anfitrión ECI** subsiguiente (en caso de un cargador por etapas) realizada mediante software procedente de una imagen anterior, utilizará para la verificación la misma clave pública de **Certificado de Anfitrión ECI** y la misma **Lista de Revocación**.

Se recomienda que los cargadores que funcionan por etapas utilicen un único mecanismo seguro para validar las imágenes de **Anfitrión ECI,** que también se utilizará para validar la primera imagen de **Anfitrión ECI** cargada.

11.3 Reglas del cargador de cliente

El **Cargador de Cliente ECI** existe en el contexto del **Anfitrión ECI**. El **Anfitrión ECI** fija la versión mínima de la Raíz **ECI** y de la **Lista de Revocación** Raíz **ECI** que utiliza para validar **Cadenas de Certificados** antes de cargar cualquier elemento relacionado con un cliente. El **Cargador de Cliente ECI** cumplirá las reglas siguientes:

1) Si es necesario la imagen de **Cliente ECI** se desencriptará en primer lugar, tal como se define en la cláusula 11.5.

2) La imagen de **Cliente ECI** y la POPK serán validadas utilizando las cadenas procesadas por el CPS como se define en la cláusula 10. Esta regla requiere una **Robustez** superior.

3) El **Certificado** de la Imagen de **Cliente ECI** o de series imágenes de Clientes (según proceda) será verificada mediante la POPK y mediante la **Lista de Revocación** del cliente **Operación de Plataforma**. La adecuación del número de versión de esta **Lista de Revocación** es verificada posteriormente por el **Cliente ECI** durante la inicialización de la sesión del **Intervalo AS**.

4) No se podrá modificar o visualizar una imagen de **Cliente ECI** una vez que ha sido cargada.

5) Los **Clientes ECI** no podrán "romper su mecanismo de seguridad (sand-box)" y visualizar o modificar el comportamiento del **Anfitrión ECI** o del **Cliente ECI**.

11.4 Aplicación de la revocación

La **ECI** utiliza un mecanismo de aplicación robusto para verificar las credenciales de las imágenes del **Anfitrión ECI** y del **Cliente ECI**. Funciona sobre la base de las reglas siguientes:

1) El desaleatorizador detendrá su funcionamiento si la versión de la Raíz **ECI** y el número de versión mínimo de la **Lista de Revocación** raíz para la verificación de la **Cadena de Certificados del Anfitrión ECI** es inferior a las cargadas por el **Anfitrión ECI** durante la inicialización. Esta regla requiere una Robustez superior.

   NOTA 1 – Esto debería de ser infrecuente ya que las **Listas de Revocación** Raíz del **Anfitrión ECI** deben actualizarse regularmente a través de los canales de todos los **Operadores**, y el **Cargador del Anfitrión ECI** puede utilizar la última **Lista de Revocación** Raíz del **Anfitrión ECI**.

2) El **Sistema AS** rechazará cargar cualquier **Cliente ECI** cuya **Cadena de Certificados** no pueda ser validada utilizando la versión de la Raíz **ECI** y el número mínimo de la **Lista de Revocación** Raíz **ECI** fijado por el **Anfitrión ECI** durante la inicialización, tal como se define en la cláusula 11.2. Esta regla requiere una **Robustez** superior.

3) El **Intervalo AS** rechazará el cálculo de las claves si el número mínimo de versión de la raíz y el número mínimo de la **Lista de Revocación** raíz que necesita el **Cliente ECI** son inferiores a los cargados por el **Anfitrión ECI** durante la inicialización. Esto se define en las reglas de cálculo aplicables a la imagen de cliente, y las claves de encriptación y desencriptación en la cláusula 8.2.4. Esta regla requiere una **Robustez** superior.

   NOTA 2 – Estas reglas garantizan que los sistemas de seguridad del contenido puedan requerir la aplicación de un estado mínimo de la raíz **ECI** para verificar todos los elementos cargados en un **Anfitrión ECI** antes de realizar alguna operación sensible en materia de seguridad.

11.5 Desencriptación de la imagen de cliente

Para desencriptar una imagen de **Cliente ECI**, el **Sistema de Seguridad Avanzada** puede desencriptar la clave de desencriptación de la imagen encriptada proporcionada por el **Operador** del **Cliente ECI** y desencriptar una imagen de **Cliente ECI** tal como se define en la cláusula 7.8 de [UIT-T J.1012]. La desencriptación de la imagen de **Cliente ECI** se realizará antes de verificar la firma de la imagen de **Cliente ECI**. La función AS utilizada para calcular la clave de desencriptación de la imagen es reqAsComputeImageKey, tal como se describe en la cláusula 8.2.4.12. El **Anfitrión ECI** recibe del **Operador** la información necesaria sobre la clave encriptada inputV (mensaje de entrada al **Mecanismo de Autenticación** a partir del cual se calcula una clave de autenticación), eKey (clave de desencriptación de la imagen encriptada con la clave de autenticación) y los parámetros "online" y "min_root_state" definidos en la cláusula 7.8 de [UIT-T J.1012], y utiliza el **Intervalo AS** proporcionado más recientemente al Cliente para la desencriptación (véase la cláusula 7.8 de [UIT-T J.1012]). Se renovarán todas las palabras aleatorias de un único uso ("nonce") utilizadas para la sesión de intercambio de claves de desencriptación de imagen (valor del **Intervalo AS** reinicializado).

12 Requisitos de temporización

12.1 Introducción

Los **Clientes ECI** funcionan en el marco de determinadas restricciones en materia de temporización a fin de cumplir los requisitos del sistema de seguridad del que forman parte. A tal fin, los **Clientes ECI** dependen de ciertas características de calidad de funcionamiento de las funciones que ofrece el **Sistema AS** (a través del **Anfitrión ECI**). En esta cláusula se define la caracterización temporal de las funciones del **Sistema AS**.

La caracterización temporal del **Sistema AS** divide las funciones en cuatro categorías:

1) Funciones que requieren simplemente *funciones administrativas* en el **Intervalo AS**.
2) Funciones que sólo requieren *operaciones criptográficas simétricas*, como los cálculos de la **Escalera de Claves** o la desencriptación con AK.
3) Funciones que requieren de una a cuatro *operaciones criptográficas asimétricas* en el **Bloque de Escalera de Claves** o en el CPS, como por ejemplo, la carga de LK1 y las funciones operacionales que conllevan el cálculo de AK.
4) Funciones que requieren el procesamiento de **Cadenas de Certificados** potencialmente más largas como las cadenas de importación/exportación y las cadenas de autenticación de **Microclientes**.

El **Cliente ECI** puede invocar funciones de las tres últimas categorías mediante mensajes asíncronos. Las funciones de la primera categoría pueden ser síncronas o asíncronas.

Las operaciones de criptografía asimétrica consumen más tiempo. Ninguna operación de criptografía asimétrica detendrá las funciones de las dos primeras categorías. Si una función de la categoría 1 ó 2 requiere el resultado de una operación de las funciones 3 ó 4, el **Cliente ECI** se responsabiliza de la sincronización del resultado de la función de categoría 3 ó 4. Es decir, tiene que esperar hasta que esté disponible el resultado de la operación asimétrica (es decir, hasta que se haya recibido el mensaje con el resultado) antes de invocar una función que depende del resultado.

12.2 Funciones administrativas

Para las funciones de la categoría 1), se aplicará el criterio general para mensajes simétricos y asimétricos.

12.3 Funciones de criptografía simétrica

Las funciones que invoquen operaciones de criptografía simétrica serán ejecutadas por el **Sistema AS** de modo que cada **sesión del intervalo AS** será capaz de realizar una función en un determinado período de tiempo. En la cláusula 6.6.3 de [b-UIT-T J Supl. 7] se propone un valor.

12.4 Funciones de criptografía asimétrica

Las funciones que invoquen operaciones de criptografía asimétrica (por ejemplo, que impliquen cálculos de claves simétricas de la **Escalera de Claves** o la utilización del resultado del **Mecanismo de Autenticación**) serán realizadas por el **Sistema AS** de modo que cada **Intervalo AS** será capaz de realizar una función a la vez. En la cláusula 6.6.4 de [b-UIT-T J Supl. 7] se proponen valores característicos.

Anexo A

Definiciones de la función criptográfica

(Este anexo es parte integrante de la presente Recomendación.)

A.1 Función Hash

Todas las funciones hash de la presente Recomendación están basadas en SHA256 tal como se define en NIST FIPS PUB 180-4 [NIST FIPS 180-4].

La cláusula 7.2 de la función *hash* definida en [NIST FIPS 180-4] corresponde a SHA-256().

La función en notación-C asHash(uchar *data, uint datalength, resultLength, uchar *result) utiliza los octetos comenzando con los datos de longitud dataLength como cadena de octetos *dataIn* y calcula la cadena de octetos resultOut como una cadena de octetos resultLength/8, que almacena como resultado con arreglo a:

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

resultLength es múltiplo de 8. truncate es la función que trunca a la izquierda la cadena de bits (parámetro 1) a una longitud de (parámetro 2) bits.

BS2OSP y OS2BSP son funciones que convierten una cadena de bits en una cadena de octetos y viceversa según se define en la cláusula 9 de [UIT-T J.1015].

A.2 Criptografía asimétrica

Las operaciones de encriptación y desencriptación asimétrica se definen en las cláusulas 10.2 y 10.3 de [UIT-T J.1015].

A.3 Generación de números aleatorios

La generación de un número aleatorio definido en la presente Recomendación cumplirá lo establecido en [NIST 800-90Ar1] así como las reglas siguientes:

- Como mínimo, en el arranque del sistema (reinicio de un **Sistema AS** basado en un circuito integrado) se generará un nuevo número semilla único aleatorio secreto. El proceso depende de aspectos físicos (ruido) u otras propiedades del circuito integrado o de su entorno que no son replicables y no pueden ser alterados. La entropía del número generado tendrá al menos 128 bits.
- Todos los números aleatorios serán generados con un generador de números pseudoaleatorios determinístico basado en el número semilla anterior de conformidad con [NIST 800-90Ar1]. El circuito integrado puede aportar regularmente nuevos números semilla a su generador y/o incrementar la entropía tal como se define en la cláusula 8.7 de [NIST 800-90Ar1], utilizando entradas internas (ruido) o externas de difícil manipulación. El id del circuito integrado se utilizará como mínimo en una cadena de personalización.

NOTA – En muchas aplicaciones AS la aleatoriedad real del generador de números aleatorios no es crítica, sólo lo es su unicidad a lo largo del tiempo. Se trata de aplicaciones típicas de números aleatorios de un solo uso, por ejemplo, un número aleatorio para la autenticación en línea a fin de evitar la reproducción en la desencriptación y la inserción de un número aleatorio en la encriptación del contenido. Una excepción es la clave aleatoria generada como LK1 en un **Intervalo AS** en modo **Microservidor** asimétrico.

Apéndice I

Ejemplo de aplicación de sistema MicroDRM

(Este apéndice no es parte integrante de la presente Recomendación.)

I.1 Introducción

Este apéndice presenta un ejemplo real de una aplicación del **Sistema AS** para implementar un sistema microDRM que funcione sobre un flujo TS. Se describe el funcionamiento en el ejemplo de **Clientes ECI** de encriptación y desencriptación. Se hace énfasis en la simultaneidad de varias acciones y la secuencia de palabras de control y mensajes microDRM asociados (del **Microservidor** al **Microcliente** y viceversa) que deben generarse. El **sistema MicroDRM** utiliza tanto la generación de una clave aleatoria en la encriptación como la generación de una palabra aleatoria de un solo uso ("nonce") en la desencriptación (a fin de evitar la reproducción). Se asume que ambas claves aleatorias tienen un límite.

I.2 Escenario de aplicación

El escenario de aplicación de la Figura I.1 muestra el estado de la **Escalera de Claves** en el lado de encriptación. LK_n es la tercera clave, que ocupa la posición más baja en la jerarquía de claves. Por debajo de ella están la palabra aleatoria de un solo uso o palabra de ocasión ("nonce") (N1 o N2) del **Microcliente**, las **Propiedades del Contenido** CP1 y CP2 (procesadas en input-C para la **Escalera de Claves** en el paso n+2) y la semilla de clave aleatoria R1 y R2 que constituye la entrada a la etapa n+3 de la **Escalera de Claves**. A partir de esas entradas a la **Escalera de Claves** se calculan las palabras de control Cw1. CW4 que se aplican al contenido conjuntamente con sus **Propiedades del Contenido** asociadas.

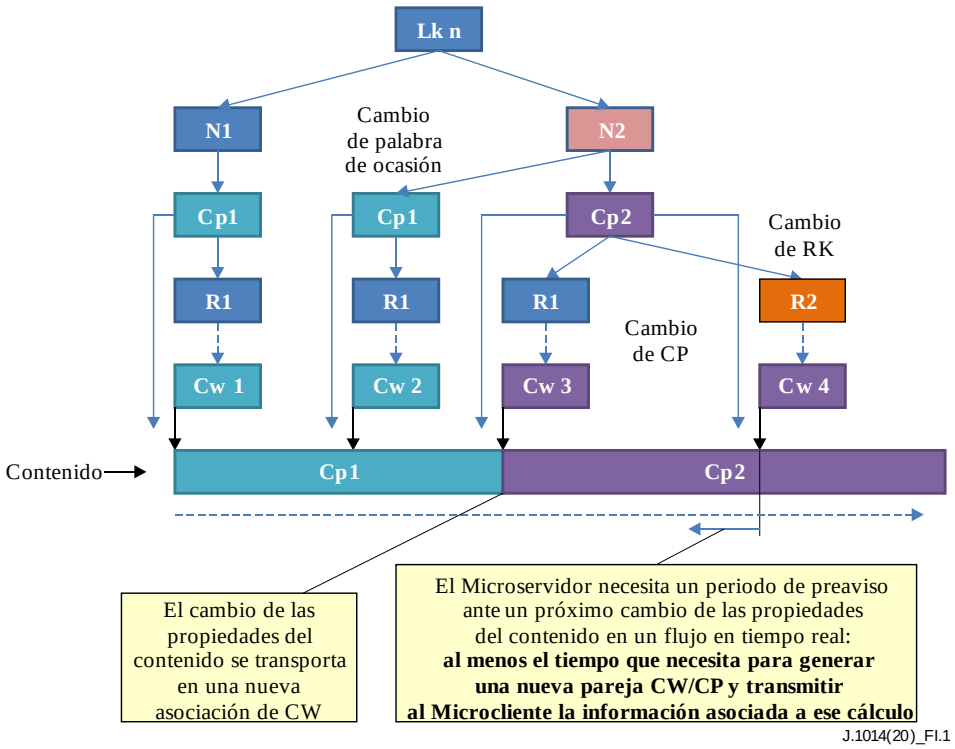

Figura I.1 – Ejemplo de evolución de la jerarquía de claves de cálculo de palabras de control

El estado inicial de las tres etapas más bajas de la jerarquía de claves del **Microservidor** es N1, CP1 y R1. A partir de ellas se calcula CW1 para encriptar el contenido. El estado inicial del bit de conmutación es t1. En este ejemplo, el **Microservidor** recibe en primer lugar una nueva palabra de ocasión y decide que es el momento de aplicarla a un próximo contenido como CW2. En primer lugar, envía un mensaje tipo ECM al **Microcliente** con el nuevo bit de conmutación t2 y el conjunto de claves encriptadas (t2, N2, CP1, R1), espera algún tiempo para estar seguro de que el **Microcliente** puede recibir y precalcular la nueva palabra de control CW2 y queda preparado a la espera del futuro cambio. Calcula entonces la nueva palabra de control CW2 y confirma su aplicación, basculando el bit de conmutación del flujo TS encriptado asociado.

El siguiente evento de la Figura I.1 muestra un cambio en las **Propiedades del Contenido** del contenido a encriptar. El **Microservidor** recibe un mensaje del **Anfitrión ECI** que informa del cambio a CP2 de las **Propiedades del Contenido**. El **Microservidor** envía un mensaje de tipo ECM a su **Microcliente** con el nuevo conjunto de claves encriptadas (t3, N2, CP2, R1) y precalcula CW3 a partir de esta nueva secuencia de claves. En el momento en que se aplican las nuevas **Propiedades del Contenido**, el bit de conmutación sobre el contenido encriptado bascula automáticamente y se aplican de manera efectiva la nueva palabra de control de encriptación CW3 y sus **Propiedades del Contenido** asociadas CP2.

El último evento es la decisión del **Microservidor** de cambiar la clave aleatoria de R1 a R2. El proceso es prácticamente idéntico al cambio de la palabra de ocasión. El **Microservidor** envía un mensaje de tipo ECM con (t4, N2, CP2, R2) al **Microcliente,** lo que le permite precalcular CW4 y aplica una demora que permite al **Microcliente** disponer de tiempo suficiente para estar preparado. Utiliza entonces la **Escalera de Claves** para calcular CW4 y la aplica al contenido, haciendo que el estado del bit de conmutación de contenido pase a ser t4.

I.3 Supuestos y notación

Un cliente exportador y el **Intervalo AS** de desencriptación asociado suministran el contenido a la conexión de importación del **Intervalo AS** de encriptación. El cliente exportador genera mensajes al **Anfitrión ECI** en los que se informa de cualquier cambio en las **Propiedades del Contenido** antes de su ocurrencia real en el contenido importado. Para ello se utiliza la API de AS de [UIT-T J.1012].

Se utiliza la notación siguiente:

<event-name>(parameters) -> <pseudo-code statement>; indica la ejecución del pseudocódigo indicado cuando se produce un evento "event-name" (recepción de un mensaje).

Se definen los eventos siguientes:

- **e_cp(cp)**: se utilizarán nuevas propiedades del contenido cp en un próximo evento (cambio de CW) que afecta al contenido a encriptar. Precede a `e_cpe()`.
- **e_cpe()**: el cambio de propiedades del contenido es inminente (dentro de un plazo de tiempo limitado).
- **e_cpch()**: las propiedades del contenido del contenido importado acaban de modificarse, siendo que la palabra de control hasta entonces utilizada no refleja la necesidad de un cambio urgente. Cuando se produce un cambio automático de la palabra de control debido a un cambio en las propiedades del contenido este evento precede a `e_cw()`.
- **e_nn(nonce)**: se ha recibido en el **Microservidor** un nuevo mensaje relativo a la palabra de ocasión ("nonce") procedente del **Microcliente** o bien, ha sido enviado desde el **Microcliente**.
- **e_cw()**: el bit de conmutación ha cambiado en el contenido reencriptado, aplicándose al contenido la nueva CW (previamente calculada).

- **`e_ecm(<parameters>)`**: recepción de un mensaje con los nuevos parámetros de la siguiente palabra de control a utilizar.
- Los eventos pueden producirse por el vencimiento de los temporizadores.

`cw(toggle_bit,random_key,nonce,content_properties)` genera una palabra de control para encriptación o desencriptación de contenido utilizando los parámetros designados. En el **Microservidor** se genera un primer mensaje con los mismos parámetros que se envía al **Microcliente**, donde se recibe como `e_ecm(…)`.

`block_cpch() and unblock_cpch()` utiliza el mensaje setAsPermitCPChange(…) para bloquear o desbloquear cambios automáticos de la palabra de control de encriptación debidos a cambios en las **Propiedades del Contenido** del contenido importado.

`changeCw(toggleBit)` obliga a un cambio de la palabra de control (bit de conmutación del campo control de aleatorización) en el lado de encriptación utilizando el mensaje setAsSC() tal como se define en la cláusula 9.9.

`startTimer(timerHandle)` arranca un temporizador.

Para las variables y el pseudocódigo se utiliza notación del lenguaje C.

I.4 Pseudocódigo del Microservidor

Se produce una dificultad en la gestión de la clave en el **Microservidor** cuando este tiene que manejar varios eventos simultáneos no sincronizados que pueden provocar un cambio de palabra de control:

- la recepción de contenido con nuevas propiedades del contenido (delimitadas por la aplicación de una nueva palabra de control por el cliente exportador al contenido a desencriptar);
- el próximo vencimiento de la palabra de ocasión; y
- el próximo vencimiento de la clave aleatoria.

El procesamiento del cambio de las **Propiedades del Contenido** debe ser prioritario ya que normalmente existe un tiempo limitado para que el cambio de la palabra de control del proceso de desencriptación active la aplicación de las nuevas propiedades del contenido. Por ese motivo, los tiempos de vencimiento de la palabra de ocasión y de la clave aleatoria deben ser suficientemente conservadores ya que su aplicación puede verse demorada por la necesidad de procesar el cambio de las propiedades del contenido (normalmente unos pocos segundos). Se asume que el tiempo entre cambios de las propiedades del contenido siempre es suficiente para permitir el procesamiento necesario para, al menos, el cambio de una palabra de ocasión o la clave aleatoria de una palabra de control.

El procesamiento de cambios importantes en una palabra de ocasión o una clave aleatoria tiene dos prioridades. En primer lugar, se fija un temporizador para la prioridad baja. Si no hay ningún cambio pendiente en las **Propiedades del Contenido** se modifica la palabra de ocasión o la clave aleatoria, en caso contrario, se fija un temporizador para un cambio de alta prioridad. Un cambio de alta prioridad de la palabra de ocasión o de la clave aleatoria puede impedir un cambio importante de las propiedades del contenido. No obstante, el **Microservidor** puede llegar a una conclusión errónea. Si así ocurre, el cambio en las propiedades del contenido se ha producido antes aplicar el cambio de palabra de ocasión o de clave aleatoria al contenido. En ese caso, debe recalcularse una nueva palabra de control que tenga en cuenta las nuevas **Propiedades del Contenido**. Asimismo, puede producirse un cambio en las propiedades del contenido casi inmediatamente después de la aplicación de un cambio de alta prioridad de la palabra de ocasión o la clave aleatoria. En ese caso, se producirán con retraso la palabra de control (CW) que refleja las nuevas **Propiedades del Contenido** y el ECM que elabora el **Microservidor**.

Si los valores de los temporizadores TNONCEURGENT y TRKURGENT pueden fijarse por encima de 10 segundos más el TECM y el tiempo máximo entre e_cpch() y e_cw(CPCHANGE) es inferior a 10 segundos, pueden producirse esas colisiones, ya que cualquier cambio de RK (clave aleatoria) o palabra de ocasión puede planificarse para que tenga lugar antes del periodo de tiempo entre e_cpch() y e_cw(CPCHANGE) o para después de dicho periodo sin que sea necesario elevar la prioridad.

Obsérvese el cliente no puede realizar directamente la manipulación de las variables `rc` y `rn`, tal como se muestra más abajo, sino que esta debe realizarse utilizando funciones del **Sistema AS**.

```
/*
   Modelo con cuatro prioridades de procesamiento con un pequeño desplazamiento del tiempo de cambio
de CP en caso de ser necesaria la prioridad 4 (aquí y ahora no se prevé cambio en CP):
    1) baja prioridad en el cambio de palabra de ocasión/rk
    2) baja prioridad en el cambio de CP (cp importante, pero no se produce e_cpch())
       Se adopta cualquier cambio previo de palabra de ocasión o rk
    3) alta prioridad en el cambio de palabra de ocasión/rk; vuelve al valor anterior de CP
    4) alta prioridad en el cambio de CP; adopta cambios pendientes de palabra de ocasión/rk y de
       nuevas cp;
       encola nuevos cambios

    Es posible una optimización que planifique los cambios pendientes de palabra de ocasión o rk
inmediatamente después de un cambio de CP; ofrece una modesta mejora de calidad de funcionamiento

    Invariantes /significados de la palabra de estado:
    <x> = cp (propiedades del contenido), n (palabra de ocasión, 'nonce') o r (clave aleatoria)
    Invariante: p<x> = cambio en <x> en la siguiente CW (p = pendiente)
                       (no para <n> o <r> de baja prioridad)
                q<x> = cambio encolado para <x>, no pendiente para la siguiente CW
                hpcp = cambio de propiedad del contenido de alta prioridad (pcp || qcp)
   Durante un breve lapso de tiempo entre changeCw() y e_cw() todos los cambios se encolan.
   Este estado temporal se indica mediante dhp==v;
 */

#define TECM 3000 /* demora entre el envío de un mensaje ecm y el cambio de CW */
#define TNONCEURGENT (2*TEMC + 1000)
#define TRKURGENT (2*TECM + 1000)
#define TNONCE /* algún valor; puede determinarse de forma dinámica */
#define TRK    /* algún valor; puede determinarse de forma dinámica */

toggle(bool t) { return !t }; /* bascula entre verdadero y falso */

encryptionSession()
/* caso de cambio de rk & palabra de ocasión y cambio de cp; aviso poco fiable de cambio de cp
(prioridad con cambio de palara de ocasión/RK) */
/* primera prioridad para palabra de ocasión/rk inferior a cp, pero si es urgente es superior*/
{
  SymKey  nc, nn;   /* palabra de ocasión actual y siguiente */
  SymKey  rc, rn;   /* clave aleatoria actual y siguiente */
  SymKey  cpc, cpn; /* valor de CP actual y siguiente */
  SymKey  nt, rt;   /* valor temporal de palabra de ocasión, clave aleatoria */
  TimerHandle t_lpn, t_lpr;
              /* temporizadores para planificar cambio de palabra de ocasión y rk de baja prioridad
*/
  TimerHandle t_n, t_r;
              /* temporizadores para planificar cambio de palabra de ocasión y rk de alta prioridad
*/
  TimerHandle t_ecm_n1, t_ecm_r1;
              /* temporizadores ecm para baja prioridad (1) ecm de palabra de ocasión y rk */
  Timerhandle t_ecm0, t_ecm1 t_ecm2, t_ecm3;
  TimerHandle t_ecm[4] = {t_ecm0, t_ecm1, t_ecm2, t_ecm3 };
              /* conjunto de temporizadores ecm con cuatro niveles de prioridad 2/3/4 */
  int t_ecm_cnt = 0;  /* contador para la asignación del conjunto anterior de temporizadores */
  bool pn, pr, pcp;   /* es verdadero si la CW actual refleja un cambio de la palabra de ocasión
                         (nn), clave aleatoria (rn) o cp (cpn) */
  bool qn, qr, qcp; /* verdadero si hay un cambio encolado de palabra de ocasión, clave aleatoria o
                       cp */
  bool dhp;         /* demora (encola) cualquier nuevo evento */
  book hpcp;        /* verdadero si la prioridad es 4: cambio de CP con alta prioridad */
  int  tCnt1, tCnt234;  /* tCnt<n> es el contador del número de temporizadores con prioridad
                           <n> descartados que aún no han vencido */
  bool t;           /* bit de conmutación */

/* se definen algunas macros para permitir la reutilización de código de procesamiento de eventos */

.* evento para la siguiente clave aleatoria */
#define next_r() { rc = rn; rn = rnd128(); startTimer(t_lpr,TRK); }
```

```
/* fuerza changeCw en el último temporizador de alta prioridad en cascada salvo que sea un cambio de
cp de prioridad 2 en cuyo caso el cambio de CW será activado por un evento de cambio de CP */
#define process_emc2_timer(){\
  if (--tCnt234 == 0)\
    if (pn || pr || hpcp){\
      dhp = true; changeCw(toggle(t));\
    } else {\
      /* pcp == verdadero, pn, pr, hpcp == falso */\
      unblock_cpch();\
    };\
}

/* con el cambio de cw se actualiza el estado con todos los cambios procesados */
#define end_pending() {\
  t = toggle(t);\
  if (pcp) { cpc = cpn; pcp = false };\
  if (pn)  { nc = nn; pn = false };\
  if (pr)  { next_r(); pr = false };\
}

/* pasa los eventos encolados a pendientes */
#define queued_to_pending() {\
  if (qcp && (!(qn || qr) || cphp)) {
    /* si la prioridad es 2 ó 4 */\
    pcp = true; qcp = false\
  };\
  /* los eventos de prioridad 3 pueden fusionarse con los de prioridad 4 */
  if (qn)  { pn = true; qn = false };\
  if (qr)  { pr = true; qr = false };\
}

/* arranca cw/ecm para cambios pendientes de 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 ;\
  }\
}


  /* solo permite autocambios del bit de conmutación cuando está preparado */
  block_cpch();

  /* recibe los primeros valores de cp y palabra de ocasión */
  for (int i=0; i<2;) {
    ->e_nn(&nc): i++;
    ->e_cp(&cpc): i++;
  }

  /* inicializa el estado */
  pn = pr = pcp = hpcp = false;
  dhp = false;
  tCnt1 = tCnt2 = 0;
  rc = rnd128(); rn = rnd128() ;
  t = false;          /* debería inicializarse al primer valor del contenido */
  cw(t,rc,nc,cpc) ;   /* comenzará a utilizarse automáticamente */

  while (!end_session) {
  ->e_nn(&nn) : startTimer(t_lpn,TNONCE) ;
                /* debe ocurrir antes de que venza el límite de la palabra de ocasión */
  ->e_cp(&cpn): /* por ejemplo, calcula nuevas licencias de exportación */ ;
  ->t_lpn()   : { /* cambio de palabra de ocasión de baja prioridad */
      if (pcp || pn || pr || cphp) {
         /* retrasa nueva palabra de ocasión hasta que sea urgente */
         startTimer(t_n,TNONCEURGENT);
      } else {
        nc = nn;
        cw(t,rc,nc,cpc);
        startTimer(t_ecm_n1,TECM) ;
        tCnt1++;
      }
```

```
    };
  ->t_lpr()   : { /* cambio de rk de baja prioridad */
      if (pcp || pn || pr || cphp) {
        /* retrasa RK hasta que sea urgente */
        startTimer(t_r,TRKURGENT);
      } else {
        next_r();
        cw(t,rc,nc,cpc);
        startTimer(t_emc_r1,TEMC);
        tCnt1++;
      }
    };
  ->t_emc_n1() :   /* vence temporizador ecm de palabra de ocasión de baja prioridad */
  ->t_emc_r1() : { /* vence temporizador ecm de rk de baja prioridad */
      if (--tCnt1 == 0 && tCnt234 == 0) {
        changeCw();
        dhp = true;
      }
    };
  ->e_cpe()   : { /* a partir de ahora puede producirse un cambio de cp */
      if (dhp || (pn || qn)) qcp = true; /* assert(!hpcp) */
      else { pcp = true; start_pending() };
    };
  ->t_n()      : { /* es necesario un cambio urgente de la palabra de ocasión */
      if (dhp || cphp) qn = true;
      else { pn = true; start_pending() };
    };
  ->t_r()      : { /* es necesario un cambio urgente de clave aleatoria */
      if (dhp || cphp) qr = true;
      else { pr = true; start_pending() };
    };
  ->e_cpch()   : { /* es necesario un cambio de CP de alta prioridad */
      cphp = true;
      if (dhp) qcp = true;
      else {
        pcp = true;
        start_pending();
      }
    };
  ->t_ecm0()   :
  ->t_ecm1()   :
  ->t_ecm2()   :
  ->t_emc3()   : {
      process_timer();
    };
  ->e_cw()     : { /* afirma ( pcp && !pn && !pr && !cphp ) */
      end_pending();
      queued_to_pending();
      start_pending();
    };
  }
}
```

NOTA 1 – En el ejemplo anterior del **Microservidor**, este puede generar y el **Microcliente** recibir varios mensajes ECM sucesivos pero diferentes con el mismo bit de conmutación. Un caso extremo es que el primer `e_n()` ocurra, y que en un tiempo `TDELAY` ocurra `e_r()` y transcurrido `TDELAY` ocurra `e_cp(..)`. En ese caso se envían tres mensajes ECM sucesivos, con el mismo bit de conmutación, y sólo el último da lugar a una palabra de control que realmente se aplica al contenido.

NOTA 2 – El código anterior asume que un cambio en las propiedades del contenido `e_cp()` siempre va seguido con relativa rapidez por un cambio en el bit de conmutación. Para el **Microservidor** no supone beneficio alguno que el nuevo valor de `cp` esté disponible mucho antes. El acontecimiento clave para generar una nueva CW es que un cambio de las `cp` en el contenido entrante sea importante. Ello provoca la sustitución del anterior valor de `cp` por el nuevo valor para todos los cálculos ulteriores de CW.

El tiempo de preaviso mínimo para la activación de `e_n()` o `e_r()` en el código del ejemplo anterior es el caso peor de la demora entre un evento `e_cp()` y el cambio real de la palabra de control subsiguiente `e_cw()`, más 2 × `TDELAY` más una cantidad menor debida a demoras de eventos y tiempo de procesamiento.

I.5 Pseudocódigo del Microcliente

El **Microcliente** comienza una sesión generando dos mensajes de palabras de ocasión sucesivos (para la palabra de ocasión actual y siguiente). Si recibe un mensaje ECM, simplemente calcula la correspondiente palabra de control. Una vez que comprueba que la última palabra de ocasión que ha enviad se ha aplicado en un ECM, genera una nueva palabra de ocasión y envía un nuevo mensaje de palabra de ocasión.

NOTA 1 – Un código de **Cliente ECI** no puede generar directamente palabras de ocasión seguras sino que debe utilizar la función adecuada del **Sistema AS**.

```
decryptionSession()
{
  Symkey nc, nn, ln;   /* palabra de ocasión actual, siguiente y última */
  SymKey cp, cpp;  /* cp recibidas y previas */
  SymKey r;        /* clave aleatoria recibida */
  bool   t;        /* bit de conmutación recibido */
  SymKey n;        /* palabra de ocasión recibida */
  bool   end_session; /* se ha llegado al final de la sesión */


  /* inicialización y envío de palabras de ocasión */
  nn = rnd128();
  e_nn(nn);
  cpp = Reserved; /* valor indefinido */
  ln  = Reserved;

  while (!end_session) {
    ->e_ecm(&t,&r&n&cp): {
        if (cp!=cpp) { /* nueva CP; envío del evento a todas las conexiones de exportación a través
                       del anfitrión */
          e_cp(cp);
          cpp = cp;
        }
        cw(t,r,n,cp);
      };
    ->e_cw(): {          /* activado también con la primera aplicación de cw */
        if (n != ln) {   /* nueva palabra de ocasión realmente utilizada; pasa a la siguiente
                         palabra de ocasión */
          nc= nn; nn= rnd128();
          e_nn(nn);
          ln = n;
        }
      };
  } /* fin del bucle 'while' */
} /* fin de la sesión de desencriptación */
```

NOTA 2 – No es necesario que el **Microservidor** devuelva al **Microcliente** los valores completos de la palabra de ocasión. En lugar de ello, y como referencia indirecta, puede utilizarse un bit alternante. Además, no es estrictamente necesario enviar todos los parámetros en todos los ECM: solo es necesario comunicar los cambios al **Microcliente**, haciendo notar que en algunos casos pueden cambiar al mismo tiempo las tres entradas a la **Escalera de Claves**, a saber, la palabra de ocasión, las **Propiedades del Contenido** y la clave aleatoria (véase la Nota 2 de la cláusula I.4). Es de utilidad enviar junto a ello un bit de conmutación a efectos de sincronización y evitar que cualquier mensaje ECM repetido (de forma deliberada o no) sea interpretado como un mensaje para el cálculo de una subsiguiente palabra de control.

I.6 Efecto en cascada de sistemas microDRM sobre la predemora del ECM

El servidor de **microDRM** utiliza un periodo de preaviso (predemora) con relación a un próximo cambio de propiedades del contenido que realiza el **Microcliente** del que importa el contenido para permitirle precalcular un mensaje ECM y enviarlo a su **Microcliente** par. El tiempo para el procesamiento necesario (que puede ser relativamente corto: normalmente no son precisos cálculos importantes) más el tiempo para enviar este mensaje ECM al **Microcliente** puede ser más largo que el necesario para el trayecto utilizado para transportar el contenido nuevamente reencriptado hasta el **Microcliente**. Eso significa que la predemora del nuevo ECM que experimenta el **Microcliente** es menor que la experimentada por el **Cliente ECI** del que originalmente se importó el contenido.

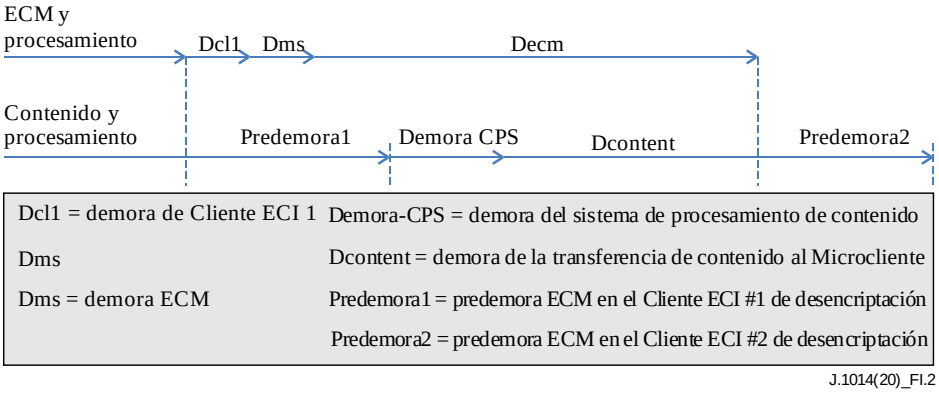

Figura I.2 – Relaciones temporales para la predemora y la compensación opcional de la demora

El **Trayecto de Vídeo Seguro** puede introducir una demora en la transferencia de contenido que compense la demora del envío de mensajes ECM, tal como se muestra en la Figura I.2. La demora puede seleccionarse de forma que sea prácticamente igual a la diferencia de retardo. Si se insertan los ECM, Decm y Dcontent quedan estrechamente relacionados. No obstante, deberían compensarse las demoras de procesamiento en el **Cliente ECI** de desencriptación y en el **Microservidor** hasta la inserción real del ECM en el flujo TS.

I.7 Convenio sobre la interfaz de temporización del cambio de las propiedades del contenido

Tal como se muestra en la cláusula I.4, el **Microservidor** precisa de un preaviso sobre un próximo cambio de las propiedades del contenido en su contenido importado. El convenio relativo al periodo de tiempo mínimo requerido para procesar el cambio y enviar un mensaje ECM al **Microcliente** se denomina TECM: en los siguientes párrafos de este Apéndice se muestra un ejemplo. En ese ejemplo, el valor de TECM se fija en 3 s.

El convenio relativo a la demora mínima de preaviso de un primer **Cliente ECI** de desencriptación en una cadena de **Clientes ECI** en cascada es TECM + TCASCADE. TCASCADE reflejan la demora máxima acumulable del procesamiento de los ECM por parte de **Clientes ECI** en una cadena en cascada de **sistemas microDRM**. En este ejemplo TCASCADE se fija en 2 s.

NOTA 1 – Si el contenido también está demorado, ello compensa las demoras de procesamiento de los ECM. No obstante, esa situación no es deseable en el modo flujo.

La demora máxima de procesamiento de los ECM de un **Cliente ECI** (Dcl1 no compensado + Dms según la cláusula I.6) se denomina TDELAY, que en el presente ejemplo se fija en 3 s. Los valores del ejemplo permiten el funcionamiento de 6 **sistemas microDRM** en cascada dentro de TCASCADE (2 s.), dejando el periodo de preaviso mínimo para el **Microservidor**.

Tal como se demuestra en la cláusula I.4, para procesar un cambio en la propiedad del contenido en el contenido entrante sin causar un desplazamiento del cambio de las propiedades del contenido, los **Microservidores** no sólo requieren un preaviso del próximo cambio del contenido, sino que también requieren un límite superior para dicho periodo de preaviso de tal modo que pueda procesar de forma segura otros cambios de palabra de control (por ejemplo, cambios de la palabra de ocasión y de la clave aleatoria). En este ejemplo, el límite superior del periodo de preaviso TMAXWARN puede ser fijado de forma segura en 10 s.

NOTA 2 – Si no se respetan estos convenios, puede producirse un desplazamiento máximo de TECM en la ubicación para un cambio de las propiedades del contenido de contenido reencriptado en un **sistema microDRM** y un desplazamiento máximo de TECM+6*TCASCADE en un conjunto de 6 **sistemas microDRM** en cascada.

Es muy recomendable diseñar los avisos de baja prioridad de la palabra de ocasión y la clave aleatoria (t_lpn y t_lpr en la cláusula B.4) con antelación suficiente para permitir que un cambio (o incluso unos pocos cambios) en las **Propiedades del Contenido** demoren el procesamiento de los cambios de palabra de ocasión y clave aleatoria. Si los cambios de las propiedades del contenido están suficientemente espaciados en el tiempo (y se respeta TMAXWARN), pueden evitarse las anulaciones del procesamiento de cambios de palabra de ocasión y de clave aleatoria.

La selección de los parámetros de temporización es importante para un traspaso sin solución de continuidad del contenido entre **Clientes ECI**. En la cláusula 6 de [b-UIT-T J Supl. 7] se proporciona más información sobre los valores propuestos de los parámetros de demora TECM, TCASCADE, TDELAY y TMAXWARN.

Apéndice II

Aspectos que se han de mejorar

(Este Apéndice no forma parte integrante de la presente Recomendación.)

Se ha llegado a la conclusión de que es necesario mejorar y validar la presente Recomendación para que cumpla los requisitos estipulados en [UIT-T J.1010] y que [UIT-T J.1010] debe actualizarse para incorporar los requisitos de la especificación de protección mejorada del contenido (ECP) de MovieLabs [b-ECP]. Las Recomendaciones [UIT-T J.1011], [UIT-T J.1012], [UIT-T J.1013], [UIT-T J.1014], [UIT-T J.1015] y [b-UIT-T J.1015.1] deben actualizarse en el futuro para incorporar esas modificaciones de la [UIT-T J.1010].

Varios Estados Miembros de la UIT y otras partes interesadas de diversas industrias – incluidos fabricantes de dispositivos y componentes electrónicos, los propietarios y titulares de contenido protegido por derecho de autor, proveedores de servicios de televisión por satélite (OTT) y de televisión por cable, y los proveedores de soluciones de sistemas de acceso condicional (CAS) y de gestión de derechos digitales (DRM) – de todo el mundo han expresado su preocupación por el hecho de que la interfaz común integrada (ECI) no satisface plenamente los requisitos de la ECP ni los requisitos más amplios de protección del contenido de la industria.

Más concretamente, sus preocupaciones se plantearon en las contribuciones a la reunión de la Comisión de Estudio 9 (CE 9) del UIT-T (16 a 23 de abril de 2020). En las contribuciones de Israel, Australia, el Miembro de Sector del UIT-T Samsung, y los Asociados de la CE 9, Sky Group y MovieLabs, se propuso la introducción de diversas modificaciones en las Recomendaciones sobre ECI, pero no se llegó a un acuerdo al respecto. Estas modificaciones se consignaron en [b-SG9 Report 17 Ann.1].

Las propuestas tienen por objeto:

1) Simplificar el sistema ECI reduciendo su alcance.
2) Eliminar el DRM.
3) Eliminar la recodificación del contenido.
4) Eliminar la gestión de software.
5) Añadir API para el almacenamiento seguro y las operaciones criptográficas.
6) Permitir escalas de claves específicas para cada proveedor.
7) Utilizar los requisitos de la norma UIT-T J.1207 TEE.
8) Incluir la implementación de la TEE para la VM.
9) Mejorar la robustez de los algoritmos criptográficos, por ejemplo, utilizando SHA-384.
10) Utilizar certificados normalizados, como los de la Recomendación UIT-T X.509.
11) Reconsiderar las comunicaciones entre clientes.
12) Realizar declaraciones de coordinación adicionales con el ETSI.
13) Realizar revisiones adicionales por pares.
14) Analizar alternativas al modelo de autoridad fiduciaria.
15) Definir más detalladamente los aspectos técnicos de las normas de cumplimiento y robustez de ECI.
16) Añadir requisitos de diversidad, por ejemplo, la aleatorización del espacio de direcciones.
17) Añadir requisitos para verificar la integridad en tiempo de ejecución.

Estas propuestas responden a que la protección de contenidos y las amenazas de su compromiso evolucionan continuamente. La ECI fue concebida originalmente casi una década antes de que se aprobara de esta Recomendación UIT-T. Los sistemas como la ECI han de evaluarse regularmente respecto del estado actual de la técnica tanto en lo relativo a las técnicas analíticas como a los requisitos de protección de la industria.

Existen otros mecanismos que permiten la interoperabilidad. En particular, en el caso de la utilización de DRM, la mayoría de los servicios de vídeo por Internet han desplegado otras soluciones que ofrecen interoperabilidad y satisfacen sus necesidades.

Es importante que haya una mayor claridad, por cuanto muchos Estados Miembros consideran que las normas de la UIT tienen gran influencia en el desarrollo de sus mercados e industrias. La lista de preocupaciones garantiza la implementación de ECI en sus mercados nacionales, lo que puede requerir comprender plenamente las repercusiones de la presente Recomendación UIT-T y velar por que se tomen en consideración dichas cuestiones cuando se examine la legislación o la reglamentación o cuando las necesidades del mercado exijan que los aparatos de televisión digital de consumo sean interoperables. También garantiza que los fabricantes de equipo tecnológico, que pueden preferir utilizar un conjunto único de requisitos u otras normas a la hora de diseñar los productos, puedan tener en cuenta estas cuestiones al desarrollar productos para diferentes mercados.

Bibliografía

[b-UIT-T J.1015.1] Recomendación UIT-T J.1015.1 (2020), *Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Sistema de seguridad avanzada – Bloque de escalera de claves: Autentificación de la información de reglas de utilización de palabra de control y datos conexos 1*.

[b-UIT-T J Supl. 7] Suplemento 7 a la serie J de Recomendaciones UIT-T (2020), *Interfaz común integrada (ECI) para soluciones CA/DRM intercambiables; Directrices para la implementación de ECI*.

[b-SG9 Report 17 Ann.1] Informe de la reunión de la CE 9 del UIT-T, SG9-R17-Anexo 1 (2020), Anexo 1 al Informe 17 de la reunión virtual de la CE 9 celebrada del 16 al 23 de abril de 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* (licencing arrangement).
http://www.etsi.org/about/what-we-do/security-algorithms-and-codes/csa-licences

[b-ETSI DVB CSA3] ETSI: *Using the DVB CSA3 algorithm* (licensing conditions).
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. Disponible en:
https://movielabs.com/ngvideo/MovieLabs_ECP_Spec_v1.2.pdf

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie D	Principios de tarificación y contabilidad y cuestiones económicas y políticas de las telecomunicaciones/TIC internacionales
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedia
Serie I	Red digital de servicios integrados
Serie J	**Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedia**
Serie K	Protección contra las interferencias
Serie L	Medio ambiente y TIC, cambio climático, ciberdesechos, eficiencia energética, construcción, instalación y protección de los cables y demás elementos de planta exterior
Serie M	Gestión de las telecomunicaciones, incluida la RGT y el mantenimiento de redes
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de la transmisión telefónica, instalaciones telefónicas y redes de líneas locales
Serie Q	Conmutación y señalización, y mediciones y pruebas asociadas
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos, comunicaciones de sistemas abiertos y seguridad
Serie Y	Infraestructura mundial de la información, aspectos del protocolo Internet, redes de próxima generación, Internet de las cosas y ciudades inteligentes
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación

Impreso en Suiza
Ginebra, 2020