国 际 电 信 联 盟

ITU-T J.1014

(04/2020)

国际电信联盟
电信标准化部门

系列J：有线网络和电视、声音节目及其他多媒体信号的传输

有条件访问和保护 – 可交换嵌入式有条件访问和数字版权管理解决方案

可交换CA/DRM解决方案的嵌入式通用接口；高级安全性 – ECI特定的功能

ITU-T J.1014 建议书

国际电信联盟

ITU-T J.1014 建议书

可交换CA/DRM解决方案的嵌入式通用接口；
高级安全性 – ECI特定的功能

摘要

ITU-T J.1014建议书是由多个部分组成的可交付成果中的一部分，涵盖有关可交换有条件访问/数字版权管理（CA/DRM）解决方案规范的**嵌入式通用接口（ECI）**之**高级安全系统**的ECI特定功能。

本ITU-T建议书是ETSI标准[b-ETSI GS ECI 001-3]的转换，是ITU-T第9研究组与ETSI ISG ECI之间合作的结果。已经对条款1、3.2、6.1、6.2、6.3和8.2.3进行了修改，进行了编辑性更正，并将“内容处理系统”一词替换为了“安全视频路径”。

沿革

版本	建议书	批准日期	研究组	唯一标识（ID）*
1.0	ITU-T J.1014	2020-04-23	9	11.1002/1000/13575

关键词

有条件访问（CA）、数字版权管理（DRM）、交换

* 为获取本建议书，请在网页浏览器内键入URLhttp://handle.itu.int/，然后输入唯一ID。例如，http://handle.itu.int/11.1002/1000/11830-en。

前言

国际电信联盟（ITU）是从事电信、信息和通信技术（ICT）领域工作的联合国专门机构。国际电信联盟电信标准化部门（ITU-T）是国际电信联盟的常设机构，负责研究技术、操作和资费问题，并且为在世界范围内实现电信标准化，发表有关上述研究项目的建议书。

每四年一届的世界电信标准化全会（WTSA）确定ITU-T各研究组的研究课题，再由各研究组制定有关这些课题的建议书。

WTSA第1号决议规定了批准建议书须遵循的程序。

属ITU-T研究范围的某些信息技术领域的必要标准，是与国际标准化组织（ISO）和国际电工技术委员会（IEC）合作制定的。

注

本建议书为简明扼要起见而使用的“主管部门”一词，既指电信主管部门，又指经认可的运营机构。

遵守本建议书的规定是以自愿为基础的，但建议书可能包含某些强制性条款（以确保例如互操作性或适用性等），只有满足所有强制性条款的规定，才能达到遵守建议书的目的。“应该”或“必须”等其它一些强制性用语及其否定形式被用于表达特定要求。使用此类用语不表示要求任何一方遵守本建议书。

知识产权

国际电联提请注意：本建议书的应用或实施可能涉及使用已申报的知识产权。国际电联对无论是其成员还是建议书制定程序之外的其它机构提出的有关已申报的知识产权的证据、有效性或适用性不表示意见。

至本建议书批准之日止，国际电联已收到实施本建议书可能需要的受专利保护的知识产权的通知。但需要提醒实施者注意的是，这可能并非最新信息，因此大力提倡他们通过下列网址查询电信标准化局（TSB）的专利数据库：http://www.itu.int/ITU-T/ipr/。

目录

引言

本ITU-T建议书[1]是ETSI标准[b-ETSI GS ECI 001-3]的转换，是ITU-T第9研究组与ETSI ISG ECI之间合作的结果。已经对条款1、3.2、6.1、6.2、6.3和8.2.3进行了修改，进行了编辑性更正，并将“内容处理系统”一词替换为了“**安全视频路径**”[2]。

本建议书的目的是促进电子通信服务、尤其是广播和视听设备市场的互操作性和竞争。然而，根据成员国的情况，还可以使用其他技术，这些技术也可能是适当的和有益的。

在迅速发展的数字广播和宽带领域，由有条件访问（CA）和数字版权管理（DRM）实现的服务和内容（包括内容、服务、网络和用户端设备 (CPE)）保护至关重要，这样才能保护内容拥有方、网络运营商和付费电视运营商的业务模式。消费者能够继续使用其已购买的CPE（如，更换网络提供商，甚或使用不同商用视频门户网站服务的装置）亦十分重要。在适当安全架构基础上，实现CPE内CA和DRM机制的互操作性，则可达成这一目标。

作为安全架构的一部分，当前建议书定义了受保护的媒体内容以及适用ECI的CPE中待处理的软件图像的鉴权和验证的安全处理系统。该安全架构的核心是由一个支持由密钥、专用芯片密钥的对象和密钥材料发端鉴权构成的安全处理的**密钥阶梯块**构建的。

第6条概述了系统架构，定义了对抗攻击的**稳健性**规定，并描述了安全架构单元、**ECI主机**和**ECI客户端**之间的关系。

第7条描述了可以采用**密钥阶梯块**的应用程序以及相关函数。

为进行正确操作，安全处理系统需要获得每个装载的**ECI客户端**的状态信息。该状态信息，由于其中的一些需要保密，将在高级安全槽的帮助下进行处理。**ECI主机**为每个**ECI客户端**分配了一个需要被保护以防被恶意篡改的插槽。对于槽的定义及其用于进行若干操作（比如解密、导出内容等）的配置的描述见第8条。

适用**ECI**的CPE内容可被解密，如果允许的话可以被转发至标准输出端，亦可以被重新加密用以导出。关于这些操作的高级安全槽的使用的规定见第9条。

以高级安全槽的特殊函数的方式实现的**证书处理子系统**负责对各项目进行鉴权。第10条规定了适用于鉴权的规定。

ECI系统采用允许**ECI客户端**安全地验证装载的**ECI主机**和**ECI客户端**凭证从而探测任何已知的安全问题的装载程序机制。装载程序机制依靠第11条描述的**稳健性**原则。

第12条包含当前建议书中描述的操作的定时限制。

---

1 附录II确定了一些有待进一步发展的领域。

2 在本建议书的文本中，使用黑体字来表示具有特定于嵌入式通用接口上下文的定义的术语，它们可能与通常用法有所不同。

ITU-T J.1014 建议书

可交换CA/DRM解决方案的嵌入式通用接口（ECI）；
高级安全性 – ECI特定的功能

1 范围

当前建议书定义了一个用于**ECI**的稳健的安全处理子系统，称为**高级安全系统**。这一**高级安全系统**为需要被鉴权和装载的软件单元提供一个安全的基础，执行安全计算和验证，管理内容以及享有相关权利和义务的内容交换的加密和解密。**高级安全系统**应用**ECI密钥阶梯块**[ITU-T J.1015]（亦可参见[b-ETSI GS ECI 001-5-2]）来执行安全计算。

2 参考文献

下列ITU-T建议书和其他参考文献的条款，在本建议书中的引用而构成本建议书的条款。在发布时，所示版本均为有效。所有建议书和其他参考文献均会得到修订；因此，本建议书的使用者应查证是否有可能使用下列建议书或其他参考文献的最新版本。当前有效的ITU-T建议书清单定期出版。本建议书引用的文件自成一体时不具备建议书的地位。

[ITU-T J.1010] ITU-T J.1010建议书（2016年），用于可交换CA/DRM解决方案的嵌入式通用接口；使用案例和要求

[ITU-T J.1011] ITU-T J.1011建议书（2016年），用于可交换CA/DRM解决方案的嵌入式通用接口；架构、定义与综述

[ITU-T J.1012] ITU-T J.1012建议书（2020年），用于可交换CA/DRM解决方案的嵌入式通用接口；CA/DRM容器、装载程序、接口、撤销。

[ITU-T J.1013] ITU-T J.1013建议书（2020年），用于可交换CA/DRM解决方案的嵌入式通用接口；虚拟机。

[ITU-T J.1015] ITU-T J.1015建议书（2020年），用于可交换CA/DRM解决方案的嵌入式通用接口；高级安全系统 – 密钥阶梯块。

[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.*（数字视频广播（DVB）；支持在数字广播系统内使用加扰和有条件访问（CA））
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.*（数字视频广播（DVB）；支持在数字广播系统内使用DVB加扰算法版本3）
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.*（数字视频广播（DVB）；使用MPEG2传输流的DVB-IPTV服务的内容加扰算法）
https://www.etsi.org/deliver/etsi_ts/103100_103199/103127/01.01.01_60/ts_103127v010101p.pdf

[ISO/IEC 9899] ISO/IEC 9899:2011, *Information technology – Programming languages – C.*（信息技术 – 编程语言 – C）
https://www.iso.org/standard/57853.html

[ISO/IEC 23001-7] ISO/IEC 23001-7:2016, *Information technology – MPEG systems technologies – Part 7: Common encryption in ISO base media file format files.*（信息技术 – MPEG系统技术 – 第7部分：ISO基本媒体文件格式文件中的通用加密）
https://www.iso.org/standard/68042.html

[ISO/IEC 23009-4] ISO/IEC 23009-4:2013, *Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 4: Segment encryption and authentication.*（信息技术 – HTTP（DASH）上的动态自适应流媒体 – 第4部分：段加密和鉴权）
https://www.iso.org/standard/62122.html

[NIST FIPS 180-4] NIST FIPS PUB 180-4:2015, *Secure Hash Standard (SHS).*（安全散列标准（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.*（NIST特殊出版物800-90A修订1：2015，使用确定性随机位生成器生成随机数的建议书）
https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final

3 定义

3.1 他处定义的术语

无。

3.2 本建议书定义的术语

本建议书定义了下列术语：

3.2.1. 高级安全系统（Advanced Security System (AS System)）：适用**ECI**的**CPE**的功能，为**ECI客户端**提供增强型安全功能（硬件和软件）。

3.2.2 AS槽（AS Slot）：由**ECI主机**排他性地提供给**ECI客户端**的高级安全块的资源。

3.2.3 AS-API：ECI客户端与其**ECI主机**之间允许**ECI客户端**与之进行信息交换并在其**AS槽**上进行操作的应用编程接口。

3.2.4 鉴权机制（Authentication Mechanism）：[ITU-T J.1015]中定义的**密钥阶梯块**函数允许一个**AS槽**出于非内容解密和加密（如鉴权）的目的提供安全密钥应用。

3.2.5 兄弟（Brother）：同一**父**的其他**子**。

注 – **父**、**子**、**兄弟**都指的是管理**证书**的**实体**。

3.2.6 证书（Certificate）：[ITU-T J.1012]第5条所定义的数据结构，亦参考[b-ETSI GS ECI 001-3]，带有识别**实体**的补充性安全数据签名。

注 – 签名的密钥的持有者通过用其密钥签署签名来证明数据的正确性—即鉴权。其公钥可被用于验证数据。

3.2.7 证书链（Certificate Chain）：互相鉴权（直到且包括**根撤销清单**）的**证书**清单。

注 – 通常，**ECI证书**中会随附一份排除了无法被验证的**证书**的**撤销清单**。

3.2.8 证书处理子系统（Certificate Processing Subsystem (CPS)）：提供**证书**验证处理并提供防止篡改的额外**稳健性**的**ECI主机**的子系统。

3.2.9 子（Child, Children）：一张由一个（共同）**父**签署的**证书**中所指的**实体**。

注 – 父、子、兄弟都指的是管理证书的实体。证书：用于启动一个CPE的SoC的初始化数据和软件。

3.2.10 内容属性（Content Properties (CP)）：提供关于与后续应用程序或内容转换相关的权利和义务的信息的内容的属性，例如使用权利信息，选择性输出控制和家长控制信息。

3.2.11 ECI（嵌入式CI）（ECI (Embedded CI)）：ETSI ISG“嵌入式CI”中规定的架构和系统，允许用户端设备（CPE）中基于软件的可交换**ECI客户端**的开发和执行，因而在**ECI**方面实现CPE的互操作性。

3.2.12 ECI客户端（嵌入式CI客户端）（ECI Client (Embedded CI Client)）：符合**ECI**规范的CA/DRM客户端的执行。

3.2.13 ECI客户端装载程序（ECI Client Loader）：使用**AS系统**来排他性地提供函数、在**ECI主机**的**ECI**容器中验证和安装新的**ECI客户端**软件图像的**ECI主机**的功能性。

3.2.14 ECI生态系统（ECI Ecosystem）：由字段中的一个**TA**和若干平台以及适用**ECI**的**CPE**构成的商业操作。

3.2.15 ECI主机（ECI Host）：一个CPE的硬件和软件系统，包含**ECI**相关功能性并拥有面向**ECI客户端**的接口。

注 – **ECI主机**是CPE固件的一部分。

3.2.16 ECI主机装载程序（ECI Host Loader）：使用**AS系统**来排他性地提供验证和安装**ECI主机软件**至一个CPE的CPE启动装载功能性。

注 – 在多级装载配置中，本术语用于描述装载**ECI主机**过程中所涉及的所有对安全至关重要的装载功能。

3.2.17 ECI根密钥（ECI Root Key）：为**ECI**认证的实体和**证书**的鉴权提供发端的公钥。

3.2.18 实体（Entity）：由**ECI生态系统**中的独一无二的ID所识别的组织（例如，制造商、**运营商**或者**安全提供商**）或现实世界中的项目（如，**ECI主机**、**平台操作**或**ECI客户端**）。

3.2.19 导出连接（Export Connection）：一个解密内容的**AS槽**和一个随后对解密的内容重新加密、表明该重新加密是被允许的**AS槽**之间鉴证的关系。

3.2.20 父（Father）：子实体的**证书**的签署者。

注 – **父、子、兄弟**皆指的是管理证书的实体。

3.2.21 密钥阶梯（Key Ladder）：[ITU-T J.1015]中所定义的**密钥阶梯块**的函数，用于计算在CPE的内容解密或重新加密函数中应用的控制字和相关控制字使用信息。

3.2.22 密钥阶梯块（Key Ladder Block）：[ITU-T J.1015]中定义的计算解密、加密和鉴权密钥稳健的安全机制，包括**密钥阶梯**和**鉴权机制。**

3.2.23 微客户端（Micro Client）：可解密被**微服务器**重新加密过的内容的**ECI客户端**或非**ECI**客户端。

3.2.24 微DRM系统（Micro DRM System）：采用**微服务器**重新加密CPE上的内容并允许被鉴权的**微客户端**解码重新加密过的内容的内容保护系统。

注 – 微服务器和微客户端由微DRM系统运营商提供。

3.2.25 微服务器（Micro Server）：可以导入解密内容、对该内容重新加密并鉴权一台特定**ECI客户端**或一组**ECI客户端**作为后续解密的**目标**的**ECI客户端。**

3.2.26 运营商（Operator）：提供被**ECI TA**征召用于签署**ECI生态系统**的**平台操作**的组织。

注 – 一个**运营商**可管理多个**平台操作。**

3.2.27 平台操作（Platform Operation (PO)）：在安全方面拥有单一**ECI**身份的技术业务提供操作的具体实例。

3.2.28 提供服务器（Provisioning Server）：服务器，通常位于一个安全的后端办公室位置，通过一个**AS槽**提供密钥和其他安全信息以助力加密或解密函数。

3.2.29 撤销（Revocation）：根据**撤销清单**中的列举而使一个**实体**被排除在外的状态。

3.2.30 撤销清单（Revocation List (RL)）：被撤销从而不得再被使用的**证书**的清单。

3.2.31 稳健性（Robustness）：一个特定的**ECI**安全函数的执行的属性，表示破解执行的安全函数的防护措施所需要的努力和/或者代价。

3.2.32 根证书（Root Certificate）：一个**ECI生态系统**中的一个证书链的鉴权的发端的受信证书。

3.2.33 安全视频路径（Secure Video Path）：所有CPE都通过微客户端或输出保护系统对内容（以及其中要求的临时存储内容）进行处理，来自并包括内容解密，来自并包括内容冲加密。

3.2.34 安全提供商（Security Vendor）：为**平台操作**的**运营商**提供包括**ECI客户端**在内的**ECI**安全系统的企业。

3.2.35 目标（Target）：一个**微客户端**或一组**微客户端**，其内容由一个**微服务器**重新加密。

4 缩写词和首字母缩略语

本建议书采用下列缩略语和首字母缩写词：

ACF 高级安全控制字段

AD 关联数据

AES 高级加密标准

AK 鉴权密钥

API 应用编程接口

ARK 高级安全随机密钥

AS 高级安全性

CA 有条件访问

CBC 加密块链接

CENC 通用加密

CISSA 通用IPTV软件导向的加扰算法

CP 内容属性

CPE 客户驻地设备

CPS 证书处理子系统

CSA 通用加扰算法

CTR 计数器模式

CW 控制字

DRM 数字权利管理

EAC 导出授权证书

EAOC 导出授权**运营商证书**

ECI 嵌入式通用接口

ECM 授权控制信息

ECP 增强型内容保护

EGC 导出组证书

ERC 导出**撤销**证书

ESC 导出系统证书

LK 链路密钥

MPEG 运动图像专家组

MSCSK 微服务器芯片组密钥

PES 封包化的的基本流

PO 平台**操作**

POC **平台操作**证书

POPK **平台操作**公钥

REAOC **撤销**导出鉴权**运营商**证书

REE 富执行环境

RFU 保留供将来使用

RK 随机密钥

RL **撤销清单**

SPK 发送方公钥

TA 可信权威机构

TEE 可信执行环境

TLS 传输层安全

TPEGC 第三方导出组证书

TS MPEG 2传输流

URI 使用权利信息

XT 延伸字段

5 惯例

在当前建议书中，使用加粗和首字母大写术语来说明这些术语采用ECI特定的含义定义，它们可能与这些术语的通常用法有所不同。

6 原则

6.1 综述

当前建议书属于ITU-T建议书系列，基于**ECI**架构[ITU-T J.1011]，亦参照[b-ETSI GS ECI 001-1]以及ECI基本要求[ITU-T J.1010]（亦参照[b-ETSI GS ECI 001-2]）。

图6-1阐述了**高级安全系统**的主要原则。**高级安全系统**的核心由[ITU-T J.1015]中定义的**密钥阶梯块**组成，允许采用密钥、特定芯片密钥的对象和密钥材料发端的鉴权的安全处理。

装载图像的基础在**装载程序核心**中体现。它采用**证书处理子系统**使用最近的**ECI根密钥**和**ECI**根**撤销清单**来验证**ECI主机**图像、**ECI客户端**图像和**平台运行**（PO）凭据的**ECI**状态。**ECI主机**和其他**ECI客户端**使用的**ECI根密钥**和**ECI**根**撤销清单**的版本号可通过装载的**ECI客户端**来校验。根据**ECI撤销**执行原则，这些客户端检测到未被接受的版本时可拒绝对内容进行解扰。加密的**ECI客户端**图像在装载时被解密。

每个**ECI客户端**使用一个高级安全槽。**AS槽**被**ECI客户端**的**平台操作**公钥识别。**ECI主机**确保通过**AS-API**进行的**ECI客户端**互动被定向到分配给该**ECI客户端**的**AS槽**。每次加密/解密操作时，每个**AS槽**通过一个槽状态和会话状态被描述。**AS槽**可以被用于解密目的或加密目的。**AS槽**会话状态亦包括一个定义操作的详细情况以及应如何对会话状态进行鉴权的配置（config）。**ECI客户端**为其他状态信息提供配置信息和输入。**密钥阶梯块**被用于对发送方公钥（SPK）、**平台操作**公钥（POPK）和配置信息进行鉴权。**AS槽**可为选择的**密钥阶梯块**输入提供随机数，以便生成随机密钥或被作为当前状态使用来确保新的计算得出的**密钥阶梯块**输入。该机制可以被用作防止加密内容的重放和确保始终通过**提供服务器**在线提供**AS槽**。

在对内容解密时，可在计算控制字的同时对**内容属性**鉴权，从而创建与解密内容相关的强链路。**内容属性**与内容一起转发至任何标准输出端，以确保妥善设定对这类输出的保护机制。这些属性与在**导出连接**上重新加密的内容的属性相比较。只允许通过适当的导出/导入**证书链**的**导出连接**。这些由**证书处理子系统**代表**AS槽**进行确认。可通过输出控制机制来禁用标准输出。

可使用交替比特协议将计算的控制字同步至MPEG传输流格式化内容。为此，**安全视频路径**使用带有用于流处理的当前/下一个控制字的双重缓存机制。

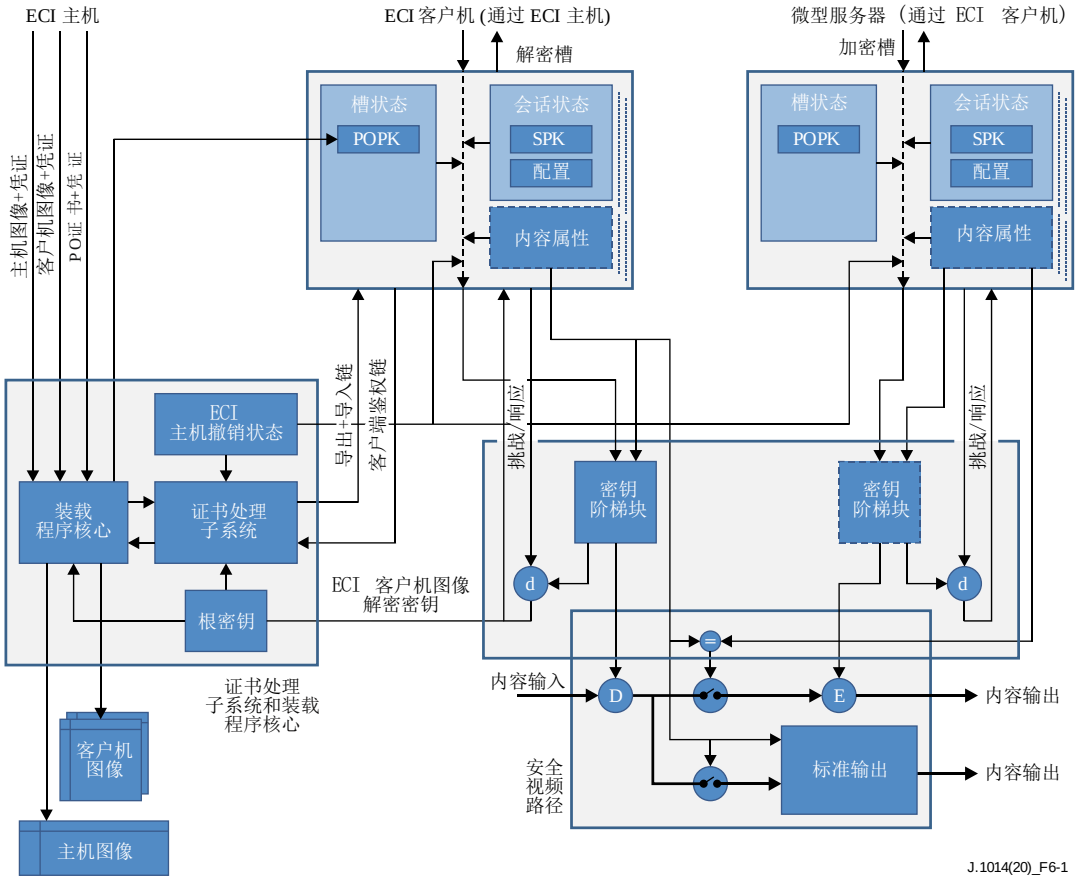


图6-1 – 高级安全系统框图

6.2 系统稳健性模型

AS系统要求稳健执行。**稳健性**通常以规避任何安全措施（如，观测一个安全系统中的保密值或操纵状态或值）所需要的努力和/或者代价来衡量。

当前建议书未为各类**ECI**功能定义特定的**稳健性**体制。不过，**ECI稳健性**架构是基于一些函数较其他函数更加稳健这一前提。图6-2对此进行了说明。

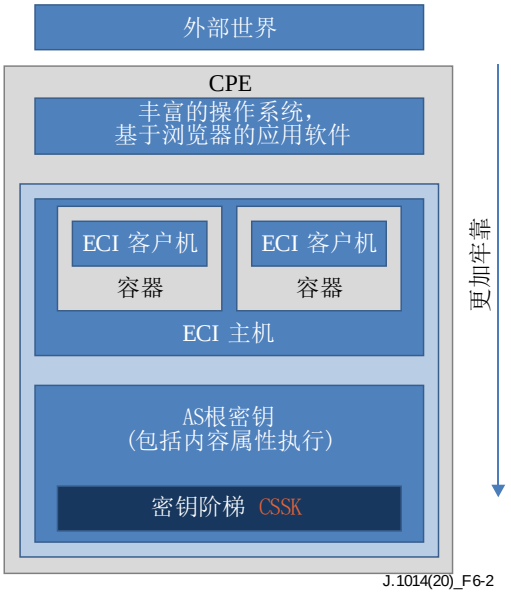


图6-2 – ECI的系统稳健性前提

最不稳健环境是任何威胁都可能存在的外部世界。需要采用鉴权和加密技术来保护通过这一环境的数据，以防它们被操纵和被未授权检测。富操作系统（通常包括一个浏览器）可多少具备防止操纵和入侵的稳健性，但通常无法承受由用户协助的或侵略性的外部黑客攻击。**ECI客户端**和**ECI主机**安全敏感性函数在一个受到良好保护、可免受这类攻击的环境中运行。若**ECI主机**被破解，**ECI客户端**亦被破解。除了对外部攻击具有恢复力之外，**ECI客户端**采用**ECI**虚拟机作为沙箱（[ITU-T J.1013]，亦参见[b-ETSI GS ECI 001-4]）：即，除了通过规定的**ECI** API接口，否则它们既无法访问**ECI主机**，亦无法访问任何其他**ECI客户端**的信息。**ECI主机**亦确保**ECI客户端**可访问**高级安全系统**和**密钥阶梯块**。**密钥阶梯块**的核心中是允许每个**ECI CPE**被独一无二地编址的**芯片组密钥**。通常**密钥阶梯块**和**高级安全系统**的主要部件是在硬件和/或者极为稳健的固件中被执行的。

在实践中，安全组件的健壮性取决于一个连续统一体，在结构上可以将它们划分为一个层次结构。在ECI背景下，这些可以如下所述：

0. 外部世界（设备的外部）的稳健性等级为0。因此，**ECI客户端**需要一个安全的协议来与头端进行通信。这超出了**ECI**建议书的讨论范围。

1. 设备的操作系统、驱动程序、应用程序和浏览器全都处于稳健性等级1。运行此代码的环境通常称为富执行环境（REE），它由大多数应用程序、操作系统系统软件和驱动程序组成。由于该软件的大小和功能很大，因此常常需要对其进行频繁修补以消除漏洞。**ECI主机**和**ECI客户端**可以使用REE提供的服务（例如，TLS网络），但不能依靠它来提供任何安全性，无论是加密还是端点鉴权。

2. 存在**ECI主机**、虚拟机和**ECI客户端**功能的执行环境，它们全部得在一个安全执行环境中运行，通常在同一个处理器上。在健壮性等级2上，这些环境仅执行经鉴权的代码、具有硬件强制的内存以及与REE隔离的设备（包括其内核和驱动程序）。这些有时被称为可信执行环境（TEE）。该等级也可以在具有内存隔离的专用处理器或安全处理器上实现。

3. **安全视频路径**要求的安全等级高于**ECI主机**和**ECI客户端**的稳健性等级3，可以结合安全硬件和安全固件来实现。

4. 最安全和最稳健的是**密钥梯度**、引导加载程序以及证书处理和撤销子系统，它们运行在稳健性等级4上。理想情况下，这些集成在安全硬件中，但由于**ECI**建议书的性质，某些部分可能需要运行于专用安全处理器上的固件。需要采取特定的措施来保护高级安全性和密钥阶梯的秘密，并进行与此相关的计算。

6.3 安全视频路径和输出保护系统控制

一旦内容被解密，**安全视频路径**就可以保护其免遭非法使用。**安全视频路径**可以对未加密的内容执行各种各样的视频（和音频）操作：多路分解、解码、缩放、重新编码、水印插入等。所有操作及其组成部分（包括所涉及的暂态存储器和用于在CPE的不同子系统之间传送内容的总线）以及控制内容属性关键处理步骤和内容输出保护设置的控制接口（内容属性和可能专用的或实施方案特定的接口）均被视为**安全视频路径**的一部分。

内容属性[ITU-T J.1012]定义了允许哪些操作以及哪些输出、输出保护系统及其设置足以保护内容。**高级安全系统**提供了一种称为“内容保护鉴权”的加密机制（参见第8.2.3条），以便在计算控制字时使用密钥阶梯块来对内容属性进行隐性鉴权。内容保护系统可以选择内容属性鉴权的设置。未由内容属性鉴权功能隐性鉴权的**内容属性**具有**ECI客户端**和**ECI主机**的健壮性等级。

应当注意，需要补充规范来提供有关**安全视频路径**和输出保护系统实施稳健性的足够详细信息。

图6-3提供了确保安全传送**内容属性**之逻辑的详细信息。首先，**ECI客户端**为下一个要解码的内容部分设置会话**内容属性**。随后，**ECI客户端**提供输入，以计算包括字段1在内的控制字（作为**elk**输入向量的一部分：参见第7.1、8.2.3、8.2.4.7条）。字段1的内容属性字段由内容属性鉴权函数中字段1的前两个字节来选择，随后使用该值作为输入到CW密钥梯度计算的对称密钥值来隐性鉴权。不正确的对称密钥值输入将导致错误的CW，因此无法解密内容。然后，SVP将字段1的值与ECI客户端设置的相应内容属性进行比较。如有任何差异，则无法输出。特定保护系统的输出也可能被相应的输出控制字段值阻止。

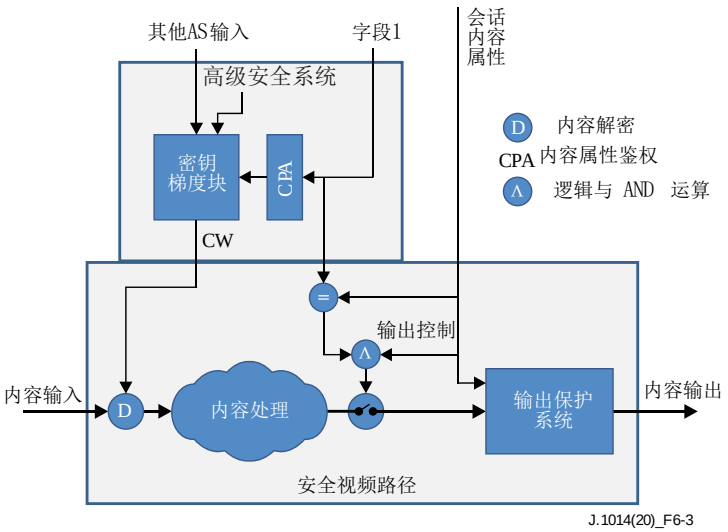


图6-3 – 安全视频路径与内容属性鉴权

6.4 规范原则

6.4.1 执行自由

当前建议书定义了在可带来新状态的**AS系统**上操作的状态和函数。一个执行状态的具体表达式并非由当前建议书定义，可由执行来完整定义，只要执行的行为可以采用当前建议书定义的转换函数被重构为状态和状态转换序列。

注 – 在许多情况下，[ITU-T J.1015]**钥阶梯**函数是状态转换函数的实质部分。

例如，一个AS执行可拥有一个可以从**平台操作证书链**对每个**密钥阶梯块**应用重新鉴权POPK的快速证书处理子系统（CPS）。在这种情况下**AS槽**无需以干预防护的方式将POPK作为鉴权值来储存。与此类似，一旦对一个**AS槽**采用两个非对称密码操作，一些执行可能决定计算LK_1（**密钥阶梯**中的顶层对称密钥），而可以足够快地执行非对称密码操作的其他执行可从其原始输入为每个**密钥阶梯**应用重新计算LK_1。

6.4.2 规范样式和与AS-API的关系

ECI客户端与**高级安全系统**之间没有直接的API。**ECI主机**被用作一个通道。不过，除slotId参数（**ECI客户端**AS API信息不需要该参数）外，一个**AS槽**上的操作的定义直接被映射到**AS API**的信息（如[ITU-T J.1012]中所定义）。**ECI主机**为**AS系统**提供slotId参数。

ECI主机（代表**ECI客户端**）在一个**AS槽**上的事务被定义为C-函数声明。这些声明描述了**AS槽**状态上的原子事务。这可导致一个新的槽状态。函数参数的具体表达式并非当前建议书规定的功能性的直接结果，除非与密码函数相关。然而，该表达式对于[ITU-T J.1012]中的**AS API**的定义至关重要。

7 密钥阶梯应用和相关函数

7.1 综述

[ITU-T J.1015]中定义的**密钥阶梯**和**鉴权机制**在一个**ECI CPE**内所有稳健执行的密钥计算中都发挥了中心作用。**高级安全系统**应采用[ITU-T J.1015]中定义的函数以及当前建议书中定义的输入和输出。所有对**密钥阶梯块**的输入都应由**AS系统**控制；根据适用的AS**稳健性**规定和**密钥阶梯块**规范[ITU-T J.1015]，不得进行任何观测或操作。

7.2 AS系统和客户端数据鉴权

高级安全系统可以与来自**ECI客户端**的数据一起提供。**AS系统**提供一种采用的**密钥阶梯块**的AD输入来验证这一数据的鉴权的方式。

AS系统计算**密钥阶梯块**的AD输入作为与CW或AK计算共同被鉴权的附加数据的散列。以下[ITU-T J.1015]AD的比特串标记应被计算为：

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

第A.1条中定义了函数的散列。lm是包含m的二进制表达式的8比特输入。m的值对应**密钥阶梯块**定义中的m的值。用于传送公钥（POPK值）的每个P_i的长度为2 048比特。C_i的长度被定义为等于第8.2.2.5小条中的SessionConfig结构的长度，XT的长度为256，适用于通用拓展机制。应被设定为0。ARK是一个128比特数，旨在表示一个随机值或在无需随机输入的情况下表示所有0。ACF是一个定义操作模式的控制值。

7.3 非对称微服务器模式

AS系统采用**鉴权机制**以装载**微服务器**发送密钥至**AS槽**用于**微服务器**和**微客户端**之间的非对称鉴权。

图7.3-1显示了非对称鉴权模式的整体计算的基本原则。

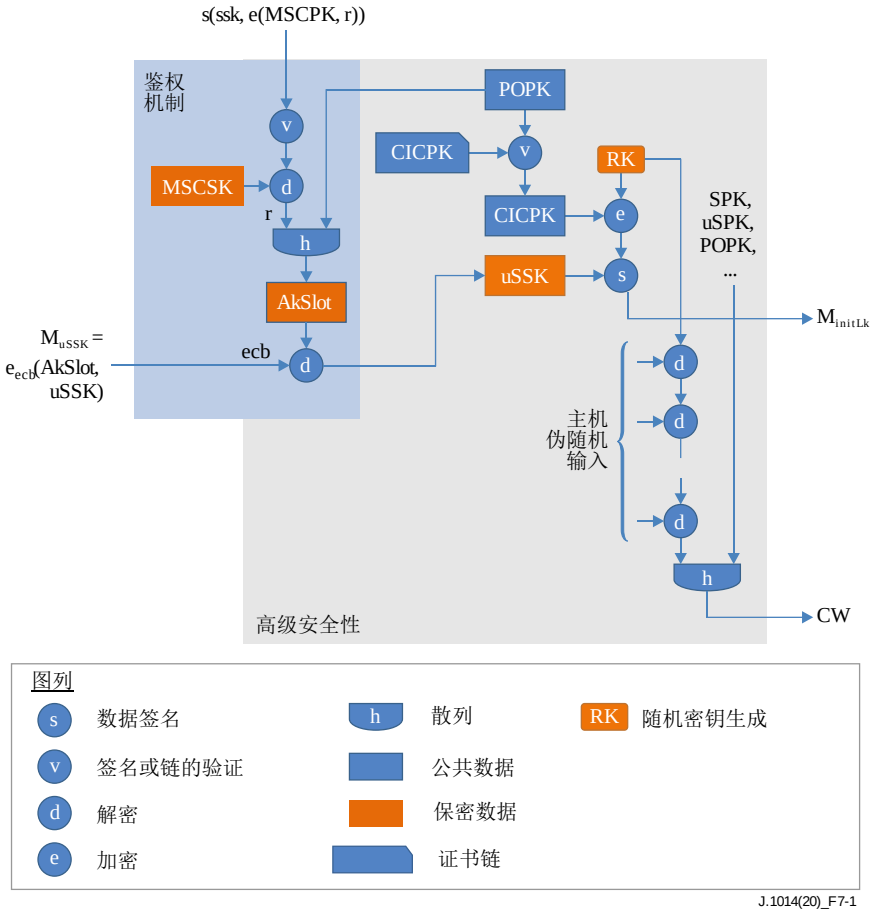


图7-1 – 非对称微服务器模式的计算

鉴权机制被用于计算**AS槽**鉴权密钥AkSlot，使用a.o.。此处的芯片组密钥被称为**微服务器**芯片组密钥（MSCSK）。AkSlot被用于装载**微服务器**密钥uSSK。**证书处理子系统**被用于给采用一个POPK为根和一个**证书链**（在链的最后一个**证书**中包含ClCPK）的**目标微客户端**的芯片组公钥ClCPK鉴权。随机密钥RK被生成，ClCPK和uSSK被用于生成**微客户端密钥阶梯**初始化信息M_{initLk}。随机密钥亦被用作一个与[ITU-T J.1015]第7.1条定义的相同结构和散列函数的**密钥阶梯**的顶层对称密钥。计算的控制字CW被用于由**微服务器**进行的加密。常规**密钥阶梯**可被用于**微客户端**中来计算用于解密的CW。

第8.2.4.10条定义了uSSK计算的特性（**ldUssk**函数）。

M_{initLk}的计算在以下计算方案中被定义，采用[ITU-T J.1015]第7.1条的规范公约：

- Mkey = cl-chipset-ID || E(ClCPK,LK)
- M_{initLk}=(Mkey || S(uSSK,Mkey))
- 其中||是按位毗连函数，cl-chipset-ID是客户端CPE的芯片组ID，E()是对称加密函数，S()是非对称签名函数，如[ITU-T J.1015]第7.2条定义。

在uSSK装载时**AS槽**应根据第A.3条生成一个新的RK。

来自LK的CW计算及其输入应与[ITU-T J.1015]第7.2条中定义的**密钥阶梯**机制一致，与该条定义的相同的输入和输出（CW, CW-URI），但将LK1的计算替换为**微服务器AS槽**生成的RK随机会话密钥。

7.4 到安全视频路径的接口

包括非对称**微服务器**模式扩展的**密钥阶梯**可计算带有补充CW-URI的控制字。这些应被安全地传送至可（暂时）储存与密钥同步信息相关的CW和CW-URI的内容处理子系统中的一个解密或加密资源。对于运输流应用，密钥同步信息由当前/下一个比特组成，因此为CW值规定两个储存地点。对于基于应用的文档，对加密和解密资源来说，单个CW地点可用。

与从**AS槽**到**安全视频路径**的控制字相关的其他信息为：

- 作为加密或解密控制字的CW应用。

注：与CW-URI一同构成应用CW的许可。

- 导出鉴权信息。
- 内容属性。
- 用于TS模式解扰的控制字奇/偶属性。

7.5 AS密钥阶梯块输入输出定义

本条定义了作为对7.2和7.3条定义的**密钥阶梯**和**鉴权机制**的输入及其扩展的表达式的C-类型变量和结构的映射。输入名的符号被用在当前建议书的其他部分以定义各种类型的应用。

C-结构到一个八比特字节序列的映射按下列（低位优先）规定定义：

- 结构的比特段按照首字段优先的顺序进行映射，从第一个八比特字节的最低比特(0)开始。
- 除了填充在下一个8比特倍数末尾（保留位应被设定为0）的8比特倍数之外的长度的结构到下一个8比特倍数。
- 16比特、32比特和64比特实体按低位优先顺序（最不重要的字节优先）映射。
- 阵列按递增索引的顺序映射。

八比特字节序列采用[ITU-T J.1015]中定义的OS2BSP函数来映射到比特串中。

注 – 上述规定确保由c-变量表示的用于整数值的比特编号顺序与[ITU-T J.1015]中与到**密钥阶梯块**对应的输入的顺序相同。

表7-1列出了变量的命名常规。

表7-1 – 密钥阶梯接口C-变量命名常规

密钥阶梯块 输入或输出	比特	C-变量命名常规
CW-URI	64	ulongcwUri;
AD	256	ucharad[32];/* 不直接使用，见第7.2 */条
SPK-URI	64	ulongspkUri;
SPK_i, i=1..16	2 048 × 16	typdefucharPubKey[256];PubKeyspk[16];/* (spk[i-1] == SPK_i) */
m		ucharnSpk;
到V的输出	64+ 2 048 + 2 048	typedef struct InputV{ ulongchipsetId; uchar elk1[256]; uchar signature[256]; } InputV; InputVinputV;
$E(LK_i,LK_{i+1})$, $i=1..24;LK_{t+1}=r$	256 × 24	typedef ucharSymKey[32];Symkeyelk[i];/* (elk[i-1] == $E(LK_i,LK_{i+1})$) */ /* C-input == $E(LK_{t-1},LK_{t1})$, i.e. the one but last input */
t		ucharnElk;
τ_b		值被设定为 0
Chipset-ID	64	ulongchipsetId;
挑战	128	ucharchallenge[16];
响应	128	ucharresponse[16];
当前文档中定义的输入和输出定义如下		
ACF	128-8	ucharacf[15];/* 操作模式 */
ARK	128	ucharark[16];
P_i	2 048 × 32	PubKeypk[32];/* 应用首个m值 */
C_i	sizeof(SessionConfig) × 32	SessionConfigconfig[?];/* SessionConfig定义见 8.2.2.6条*/
XT	256	ucharXT[32];/*值始终被设定为 0 */
M_{initLk}	64+ 2 048 + 2 048	InputVmInitLk;

以下c-函数采用上述输入变量进行定义来产生结果。

```
SymKeyblockV_blockC_KeyLadder(InputVinputV, SymKeyspk)
```

语义：

这一函数计算**密钥阶梯**中的V块和C块的函数来产生lk1。

在使用非对称服务器情况下，以下函数计算**目标微客户端**的初始信息，如第7.3条条定义：

```
InputVasymInitLk1(SymKeylk1, PrivKeyussk, PubKeyspk);
```

语义：

这一函数根据第第7.3条条计算初始信息initLk1。

密钥阶梯的剩余函数由以下执行：

```
keyLadder(SymKeylk1, ulongcwUri, ucharacf[15], ucharark[16],
    PubKeypopk[16], SSCnfgclCnf[16], ucharXT[32], ulongspkUri, uintnSpk, PubKeyspk[16], ucharnElk,
SymKeyelk[32])
```

语义：

这一函数采用lk1作为**密钥阶梯**中的D块的结果来计算**密钥阶梯**的余数，从而生成CW结果用于**安全视频路径**。

计算AK密钥的**鉴权机制**的函数由以下执行：

```
SymKeyAuthMech(InputVinputV, ucharacf[15], ucharark[16],
    PubKeypk[16], SSCnfgclCnf[16], char XT[32], ulongspkUri, uintnSpk,
    uintspkIndx, PubKeyspk[16])
```

语义：

这一函数计算上至AK的**鉴权机制**并提供结果。

为了使用计算出的AK密钥，采用[ITU-T J.1015]第8条的**鉴权机制**中的挑战—响应接口和函数块d来定义以下函数。

```
uchar[16] AuthMechResponse(SymKeyak, uchar[16] challenge)
```

语义：

这一函数采用AK作为**鉴权机制**中定义的鉴权密钥计算挑战输入的响应。

7.6 ACF定义

到**密钥阶梯块**的高级安全控制字段（ACF）输入用来定义操作模式的主要特性。表7-2规定了acf[0]参数的值。

表7-2 – 密钥阶梯应用的ACF[0]

Acf[0]	值	描述
AcfCw1模式	0x11	**密钥阶梯**操作同当前建议书定义。acf [1]..acf[14]应等于0x00。
AcfAk1模式	0x12	**鉴权机制操作**同当前建议书定义。acf[1]的值被称为**AkModeField。**表7-3定义了适用的值。Acf[2]..acf[14]应等于0x00。
保留	其他	保留供未来使用。

作为**密钥阶梯**参数的应用的AcfCw1Mode模式的补充c-定义为：

```
constuchar acfCw1Mode= { AcfCw1Mode, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
```

作为**密钥阶梯**参数的应用的AcfAk1Mode模式的补充c-定义为：

```
constuchar acfAk1Mode= { AcfAk1Mode, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
```

表7-1 – AcfAk1Mode的AkModeField定义

寄存器	比特	值		描述
AkUseFlag	8	0b0	**AkUseAS**	仅供**高级安全系统**的AK应用。
		0b1	**AkUseCl**	**ECI客户端**AK应用。
AkOnline	7	0b0	**AkOffline**	AK在一个单向“离线”模式中创建。挑战/响应可被提前计算。
		0b1	**AkOnline**	AK采用随机当前状态AKRK（要求“在线”计算挑战响应）来创建。
AkAsAppl 只有AkUseFlag= **AkUseAS** 其他情况保留	0..3	0x0	**AkConfigAuth**	鉴权**AS槽**的配置元素。
		0x1	**AkLdUssk**	使用AS来解密和装载一个**微服务器**uSSK密钥。
		0x2	**AkClImg**	使用AK来解密用于解密待装载的**ECI客户端**图像的密钥。
		0x3.. 0xF	保留	保留供未来使用。
RFU	其他	0		保留供未来使用。值应被设定为零。

8 高级安全槽

8.1 高级安全槽引言

高级安全系统包含每个装载的**ECI客户端**的状态信息。将一个**ECI客户端**与一个**AS槽**绑定的标识是**ECI客户端**的**平台操作**公钥（POPK）。**ECI主机**装载**ECI客户端**的POPK至一个可用的**AS槽**。之后**AS槽**的状态与该**ECI客户端**相关。任何**AS槽**的有意义的操作都要使用POPK作为输入，从而使结果只针对绑定的**ECI客户端**而对于其他客户端来说无意义。

ECI主机稳健性确保一台**ECI客户端**仅可访问为其指定的槽的信息：在**ECI主机**正确运行的时候可确保只有指定的**ECI客户端**才可访问自身的**AS槽**。若**ECI主机**不知何故被攻陷时，POPK的“锁定机制”确保只有可组成一个凝聚集的到**AS槽**的AS输入才能以解锁密钥（CW）和相关的**内容属性**或鉴权密钥（AK）的形式产生有意义的结果。

若**ECI主机**决定将一个**AS槽**重新设定给另一台**ECI客户端**使用，任何之前**ECI客户端**（POPK）累积的状态则被消除。

8.2 AS槽定义

8.2.1 综述

从***状态变量***和***输入变量***到***状态修改函数***等方面定义**AS槽**。本条的状态表达式、输入和输出值的表达是经过挑选的，如此在这些值上进行的操作按照在此定义的二进制表达式进行定义。这与其加密和**密钥阶梯**计算中的合并尤为相关。在实际操作中可以选择各自的状态表达式但任何自定义表达式必须转换为在此规定的表达式用于任何加密操作的输入。

一个**AS槽**的所有状态变量都应被稳健地保护，以防任何恶意篡改。一些状态变量保存仍需保密的信息：这类寄存器应被稳健地保护，不被未授权访问。这类变量采用“保密”作为C-类型定义的一部分来定义。任何基于保密变量值的计算应被保密，除非这一结果是明确共享的。任何保密值和/或者带有保密值的计算的储存地点都应拥有与**密钥阶梯块**[ITU-T J.1015]所要求的相同程度的**稳健性**。

AS槽存在会话。每个会话根据其配置（会话状态的一部分）的设定运行。会话配置由**ECI客户端**设置，使用前必须通过使用**鉴权机制**或**密钥阶梯**的隐性鉴权属性来鉴权。

所有状态变量和函数都根据C-语言[ISO/IEC 9899]来定义。C-语言的序列顺序并未严格遵守，因为项目可以在其使用之后再定义。固定大小的阵列同一个单独的赋值语句一起被复制（而非复制指针值），如同它们被包含在一个“结构体”结构中。

至于差错，通过定义隐性差错校验，使代码保持更强的可读性。为状态变量或其字段（见字段定义）指配一个保留值是一个差错。如果这类指配表达式的右侧是基于一个单一的函数参数，则会给该参数返回一个差错：参数i的值为-i。

除非另有明确规定，否则所有字段和变量的缺省值被定义为0。

8.2.2 AS槽状态定义

8.2.2.1 槽和会话状态

每进行一次加密或解密会话**AS槽**状态就被定义为一个共用槽状态和会话状态。槽状态的结构定义如下。表8-1对字段进行了定义。

```
#define NSLOTS       /* (maximum) number of slots */
#define NSESSIONS    /* (maximum) number of sessions */
#define MaxSpkEncr 4 /* maximum number of encryption SPK values */

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;
    RkStaterkState;
    importExportStateies;
} 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[NSLOTS]
```

表8-1 – AS槽状态结构定义

字段	描述
active	会话处于活跃状态为真，否则为假。缺省状态为假。
configAuthMode	槽配置已经被鉴权的模式。允许值为： ConfigAuthModeNone：0x0，槽配置已经被鉴权。 ConfigAuthModeAk1: 0x1，已使用8.2.4.8条定义的AK机制对槽配置进行鉴权。 其他值保留供未来使用
Mh	与AS槽会话相关的媒体处理。
clientCheckFlag	已装载一个新的ECI客户端。POClRLVnr的验证应在会话初始化时处理。缺省值为0b1。
Reserved	字段被保留；应被设定为零。
POClRLVnr	用于在装载前验证ECI客户端平台操作客户端撤销清单版本号。每次ECI客户端会话初始化时对照ECI客户端要求的最低版本进行检查。
slotConfig	槽配置。
spk	用于计算LK1和AK的公钥。
spkUri	用于计算LK1的SPK矢量使用规定信息寄存器。
spkIndx	用于LK1计算的SPK矢量中的指标选择SPK寄存器位置。
coupledSessionId	仅在AS槽会话处于解密模式时使用。与第二个（解密）会话耦合。解码的内容流被加入，内容属性被比较。缺省值为-1。
nEncr	用于加密的SPK/POPK输入值的数量（槽的spk除外）。
encrSpk	带有密钥阶梯的加密内容的SPK值。
encrPopk	带有密钥阶梯的加密内容的POPK值。
encrCwUri	带有密钥阶梯的加密内容的CwUri值。
lk1	采用密钥阶梯计算控制字的顶层链路密钥。

表 8-1 – AS 槽状态结构定义

字段	描述
Ussk	微服务器密钥（用于微服务器应用）。
rkState	随机密钥会话状态。
ies	导入/导出会话状态。
version	槽状态版本。容许值为： 0x1：版本1。 所有其他值保留。
slotMode	槽的操作模式。容许值为： SlotModeDecr：0x1，在解密模式中操作的槽。 SlotModeEncr：0x2，在加密模式中操作的槽。 所有其他值保留。
popk	使用该槽的ECI客户端公钥。
slotRK	用于在线挑战响应协议的随机数，即，带有提供服务器。值在槽初始化时设定。
akClient	用于客户端处理目的的鉴权密钥。
se	会话状态（针对一个AS槽中的所有会话）。
ss	AS槽状态（针对所有槽）。

除非另有说明，否则每个状态元素在初始化时的缺省值均为零。

8.2.2.2 解密配置

以下c-代码定义了解密模式中的**AS槽**会话适用的配置状态，表8-2给出了对它的描述。该表定义了在解密模式下操作的AS槽会话操作的详细信息。可采用适当的**鉴权机制**或**密钥阶梯**计算来对这些数据进行鉴权。

```
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;
```

表8-2 – DecryptConfig结构定义

字段	描述
configVersion	解密配置的版本。定义值为0x1：版本1。所有其他值保留。AS槽会话拒绝执行没有容许值的字段的任何状态转换函数。
reserved1	保留字段；应被设定为0。
klModeAuth	若设定了这一比特，AS槽会话应采用ClientConfig来鉴权所有的密钥阶梯计算。该比特本身在所有密钥阶梯计算中被鉴权。
akModeAuth	若设定了这一比特，AS槽会话应在允许任何密钥阶梯计算前验证configAuthMode被设定为ConfigAuthModeAk1。该比特本身在所有密钥阶梯计算中被鉴权。
rkKlMode	若设定了这一标记，AS槽的所有密钥阶梯计算中应采用slotRK。
spk0NoDecrypt	被设定时，不得使用spk[0]（spkIndx==0作为密钥阶梯函数的输入）用于解密模式下的槽LK1的鉴权。
reserved2	保留字段；应被设定为0。
rkDecrMode	定义用于密钥阶梯计算的随机会话密钥的应用。见第第8.2.2.5条。
minEciRootState	ECI根版本和根撤销清单版本的最小值。若CPS采用小于本结构中的值的ECI根密钥或根撤销清单用于ECI鉴权目的，本会话的密钥阶梯计算将不被允许。
minClientVersion	ECI客户端版本。用于验证POPK的撤销清单版本号。

8.2.2.3 加密配置

以下c-代码中定义了加密模式中的**AS槽**会话的适用配置状态，表8.2-3给出了对它的描述。

```
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;
```

表8-3 – EncryptConfig结构定义

字段	描述
configVersion	加密配置版本。定义值为s 0x1：版本1。所有其他值保留。AS槽会话不执行没有容许值的字段的任何状态转换函数。
Reserved1	保留字段；应被设定为0。
microServerVersion	微服务器配置的版本号。亦被用作非对称微服务器模式中微客户端鉴权的最低撤销清单版本号。
asymKlMode	若设定该标记，密钥阶梯应根据第7.3条定义的非对称客户端鉴权机制操作。
rkKlMode	若设定该标记，slotRK应被用于任何密钥阶梯计算。
Reserved2	保留字段；应被设定为0。
rkEncrMode	定义用于密钥阶梯计算的随机密钥的应用。见第8.2.2.5条。
basicUriTrfr	在应为基本URI作为加密内容的内容属性之前定义来自导入连接的基本URI的状态转换。值见表8.2-5。
contPropControl	定义如何计算加密内容的内容属性。见表8-4。
defaultCP	所有内容属性字段的一个缺省值。在密钥阶梯计算中的应用受contPropControl字段的控制。
minEciRootState	ECI根版本和根撤销清单版本的最小值。若CPS采用小于本结构中的值的ECI根密钥或根撤销清单用于ECI鉴权目的，本会话的密钥阶梯计算将不被允许。。

contPropControlFields字段是由16个2-bit字段组成的阵列。这些2-bit字段表明用于加密输出的Field1**内容属性**是如何被控制的。表8-4提供了描述。CpControlFlag bit-2n和bit 2n+1应与Field1-byte n对应。

表8-4 – CpCtrl定义

标记名	值	描述
CpCtrlCopy	0b00	应从导入连接中复制CP Field1内容属性字节
CpCtrlDef	0b01	CP Field1内容属性字节应被设定为相对应的defaultCP字节的值
CpCtrlMS	0b10	rCP Field1内容属性字节应由微服务器设定
保留	0b11	值被保留

basicUriTrfr字段在其CPControlFlag状态与**CPCopy**相等时，对BasicUri字段的CPControlFlags的上述行为进行修改。表8-5定义了替代行为。

表8-5 – BasicUriTrfr值和描述

标记名	值	描述
JustCopy	0x00	从导入连接中复制出Field1内容属性字节
NoMoreCopy	0x01	RedistributionProtected的基本URI状态应被转换至仅查看（ViewOnly）状态
Reserved	其他	保留供未来使用
注：通过将BasicUriTrfr状态设定为NoMoreCopy，微客户端系统仅允许任何受保护内容输入的流传输至微服务器。		

8.2.2.4 随机会话密钥控制

下列c-代码中定义的RKMode结构和表8-6定义了**密钥阶梯**中必须采用随机会话密钥的模式。

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

表8-6 – 解密和加密会话的随机密钥结构

字段	描述
模式	定义随机会话密钥的应用模式。值为： 0b00：RKModeNone，无插入的随机会话密钥。 0b10：RKModeDataLimit采用有数据限制的随机会话密钥。 0x11：RKModeTimeLimit采用有时间限制的随机会话密钥。 0b01：值被保留。
限制	该值从自随机密钥初始化起解密或加密的数据的实时秒或k字节方面定义了可用的限制。函数limitValue()定义了适用的实际限值。值63被保留。

```
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 总会话配置

以下规定的SessionConfig结构定义了处于加密或解密模式的一个**AS槽**会话的完整配置控制信息。若**AS槽正**处于加密模式，则包含用于后续解密的配置信息。

```
typedef struct SessionConfig {
    EncryptConfig   encryptConfig;/* configuration for encryption */
    DecryptConfig   decryptConfig;/* configuration for decryption */
} SessionConfig;
```

定义了用于验证**ECI证书链**的**ECI**根状态的cpsEciRootState结构在如下c-代码和表8-7中定义。

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

EciRootState cpsEciRootState;/* contains the minimum value from the CPS */
```

表8-7 – EciRootState结构字段描述

字段	描述
rootVersion	ECI根证书的版本
rlVersion	与根证书一同使用的撤销清单的版本
注： EciRootState通常	

以下函数被定义为检查cpsEciRootState继续计算密钥是否充分：

```
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);

  /* following should not occur */
  return false;
}
```

先决条件：

- **AS槽**slotId被初始化。

8.2.2.6 随机会话密钥状态

对于与**AS槽**相关的每个解密或加密会话来说，**AS槽**储存了以下c-代码中定义、在表8-8中规定的随机密钥状态信息。

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

表8-8 – RkState随机密钥状态字段描述

字段	描述
rkCurrent	用于密钥阶梯中的插入以计算CW的当前随机密钥
rkNext	待插入密钥阶梯以计算CW的随机密钥的下一个值。
limitCounter	表明与应用于密钥的限值相关的单元中的当前密钥的使用状态的计数器。该值基于与rkCurren一起计算的CW来计算仍可以被加密或解密的剩余单元。

limitCounter字段应在CW应用时递增。

注 – 实施方案可有效执行作为**安全视频路径**组成部分的计数器。

8.2.2.7 导入导出状态

每个加密会话都有一个与之相关的解密会话，加密会话从该解密会话中导入待被重新加密的内容。导出会话的（至少）两个导出组应能够同时导入，允许无缝转换。

两个解密会话可以耦合。允许需要其他控制字（由同一**AS槽**计算）的不同子流在转发至行业标准输出或导出之前被合并至拥有一套**内容属性**的一个合成流中。作为合并过程的一部分，**AS系统**检查合成流的**内容属性**是否相等。

注： 比较**内容属性**亦可涉及导出组id，这样确保耦合的两个会话所需的导出链处理是相等的。

这连同随机密钥状态一起构成**AS槽**中的会话绑定状态。以下c-代码定义了会话状态'se'；表8-9列出了字段描述。

```
#define MaxExpGroupIds 2

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

#define ImportNone -1
```

表8-9 – ImportExportState结构定义

字段	描述
importSlotId	仅在AS槽处于加密模式时适用。该值是从中导入内容的槽（“导入槽”）的槽数。缺省值为-1。
importSession	仅在AS槽处于加密模式时适用。该值是导入槽（从中导入内容）中的会话数。缺省值为-1。
expGrpId[eid]	仅在AS槽处于加密模式时适用。从中导入内容的导出AS槽的导出组id。值0x00被保留。
importPermitted[eid]	仅在AS槽处于加模式态时适用。若导出AS槽允许expGrpId[eid]，则设为真；否则设为假。缺省值为假。
rkState	本次会话的随机会话密钥状态。

若对应的解密会话被重置或重新初始化，则**AS系统**应移除一个导入会话（将对应的ImportPermitted字段设定为假）。在**AS槽**重置或重新初始化时**AS系统**应重置**AS槽**的所有会话。

8.2.3 内容属性鉴权

执行解密函数的**ECI客户端**通过各自内容属性API向**ECI主机**提供内容属性值。**ECI主机**应向**高级安全系统**输入这些值以及所要求的数据，以计算适用内容的控制字。**高级安全系统**应确保**内容属性**的妥善执行并通过使用它们来计算对于**AS密钥阶梯块**的C-输入来验证**内容属性**。

在计算至**密钥阶梯**的C-输入的过程，**微服务器**使用在**AS系统**或**ECI客户端**上采用与上述计算相同的机制传递或处理的**内容属性**。加密模式中使用的**AS槽**根据**微服务器**配置设定对照由解密资源转发的**内容属性**验证**微处理器**提供的**内容属性**。在探测到不匹配时停止加密。

为进行鉴权和验证，**内容属性**被分成两个阶段并入字节序列。第一阶段将较小的固定长度内容属性字节合并进*field1*。fieldControl字节控制用于鉴权的字节大小内容属性字段的呈

现。在第二阶段，更长的内容属性字段被编入字段序列*field2*。*field1*和*field2*被串接并输入至一个将所有字节压缩成为一个用于**密钥阶梯**的C-输入的128-bit值的散列函数。表8-10提供了field1结构。

表8-10 – field1结构定义

名称	类型	字节数	描述
fieldControl	字段控制	0,1	这一字段定义了最低有效位在字节0中的16比特值。见表8-11。
basicUri	字节	2	这一字段的值应对应BasicUri类型规范，[ITU-T J.1012]，表9.8.2.5.1-1。
outputControl	字节[2]	3-4	这一字段的值应对应输出控制矢量规范[ITU-T J.1012]，表9.8.2.6.1-1。
standardUri	字节[3]	5-7	这一字段的值应对应于标准URI规范[ITU-T J.1012]，表9.8.2.3.1-1。
exportGroup	字节	8	被解译为表示应用于内容的导出组id的无符号整数。值等于0应被ECI主机解译为不允许任何导出；值0x80 - 0xFF被保留。
parentalAuth	字节	9	如[ITU-T J.1012]第9.8.2.8.1-1条定义，对应ParCond.basicCondition，比特[0..5]被设定为0b000000。
保留	字节[6]	10-15	按照当前建议书要求，字节应被ECI主机设定为0x00。

表8-11 – FieldControl结构定义

名称	比特	描述
bit-<n>	2-16	这一比特控制field1的字节-<n>的确认。若值为0b1，说明字节<n>被确认，并等于所指字段。若值为0b0则说明<n>不被确认，应使用值0x00替代用于Field 1中的字节<n>。当被用作计算解密控制字的输入时，bit-2应被设定为0b1，这样确保basicUri始终会对照内容加密时使用的值被鉴权。
Field2ctrl	0-1	值0b00说明field2不存在。值0b01说明字段存在并使用如下定义的编码。值0b10和0b11被保留并且不得被使用。

Field2的定义采用带有总长度字段的标签/长度/值结构来确保总体完整性。本条对Field2结构定义如下。

函数computeField1Decrypt计算用于鉴权中下一步的选择逻辑：

```
void computeField1Decrypt(uchar field1[16], uchar result1[16]) {
    int i;
    ushortfieldControl = 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;
     }
```

处于加密模式中的一个**AS槽**应计算用result1来表示的对内容属性鉴权的输入和用于比较field1字节与属于输入会话内容的field1的cpMask字段：

```
void computeField1Encrypt(
            uchar msField1[16],/* field1 for CP from Micro Server Client */
result1[16],        /* result CP for authentication in computing CW */
            ushortcpMask,               /* result mask for comparing msField1 to client's
                                        version of CP field1 */
            EncryptConfigssEncrypt  /* encryption configuration of the AS slot */
) {
    int i;
    ucharcp[16];/* CP value to be computed */
```

```
    /* set control bytes of content properties */
    cp[0] = ssEncrypt.defaultCp[0];
    cp[1] = ssEncrypt.defaultCp[1];
    mask = 0x0000;

    /* process the contPropControl rules to compute cp */
    for (i=2;i<16;i++) {
        switch (ssEncrypt.contPropControl>>(2*i) && 0b11) {
            case CpCtrlCopy:/* shall be copied from import Client */
                if (i==2) { /* basic URI byte */
                    /* process basicUriTrfr */
                    switch (ssEncrypt.basicUriTrfr) {
                    case BasicUriTrfrNoChange:
                        cp[i]= msField1[i];
                        break;
                    case BasicUriTrfrNoMoreCopy:
                        if ((clField1[2]&0b11) == RedistributionProtected)
                            cp[i]= (msField1[1] & 0xFC) + ViewOnly;
                        else
                            cp[i]= msField1[i];
                        break;
                } else {   /* all other CP bytes */
                    cp[i]= msField1[i];
                }
                cpMask += 1<<i;/* msField1 byte i to be compared to imp client */
                break ;
            case CpCtrlDef:/* shall be set to default CP from configuration *?
                cp[i] = ssEncrypt.defaultCP[i] ;
                break ;
            case CpCtrlMS:/* shall be defined my software Micro Client */
                cp[i] = msField1[i];
                break;
        }
    }

    /* compute input to authentication function same was as for decryption */
    computeField1Decrypt(cp, result1);
}
```

Field2为结构化的字节序列，定义如下：

```
typdef struct Field2 {
    uint length;        /* number of bytes in content, shall be a multiple of 4 */
    byte content[]; /* content defined below */
} Field2;
```

Field2结构的内容字段应包含LargeProperty结构（每个都带有一个独特标签）的一个序列。LargeProperty由以下c-代码定义。

```
typdef struct LargeProperty {
    uintpropertyTag;    /* see Table 8.2-12  */
    uint  length;       /* length of property field in bytes
    byte  property[];   /* contains the actual property value */
    byte  padding[];    /* additional bytes set to 0x00 to make LargeProperty a
                        multiple of 4 bytes large */
} LargeProperty;
```

largeProperty属性标签字段值和对应的属性字段定义见表8-12定义。

表8-12 – largeProperty标签字段值和含义

属性标签值	属性
0x00000000	保留
0x00000001	对应[ITU-T J.1012]第 9.8.2.7.5条定义的setDcrMarkBasic信息参数数据
0x00000002	对应[ITU-T J.1012]第 9.8.2.7.6条定义的setDcrMarkExt信息参数数据
0x00000003	第 9.8.2.4.1条定义的setDcrCustUri信息参数custURI
其他	保留供未来使用

AS系统可拒绝超过其对于*field2*处理能力的任何数据。

AS系统应采用下列检查来核对任何Field2数据参数的一致性：

- 构成LargeProperty结构的长度等于Field2结构的长度字段。
- 所有构成LargeProperty结构的填充字节为0x00。

密钥阶梯的相关数据输入C应根据以下c-代码从result1和field2计算：

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

    if (result1[0] & 0b11 == 0x00) {
        /* no field2 to be included */
        for (i=0;i<16;i++) hashIn[i] = result1[i];
        asHash(hashIn, 16, 128, input_C);
    } else if result[0] & 0b11 = 0x01) {
        /* field2 to be included for 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);
    }
}
```

其中asHash为第A.1条中定义的第一个参数中的字节序列、第二个参数的字节序列的长度、第三个参数的结果的一个比特-长度和最后一个参数中的结果的散列函数。

外散列计算（直接计算）的**稳健性**应至少与内散列的散列计算的稳健性一样高。一个散列的**稳健性**的测量反映了在散列函数的输入的任何内容与这些输入作为内容属性以及操纵散列函数和/或者其输出的应用之间制造差异所需的工作量。

两个散列计算的**稳健性**的不同级别的一个例子是外散列可以由一个稳健的硬件结构进行而内散列可以由一个稳健的软件执行来进行。

8.2.4 AS槽函数

8.2.4.1 概述

通过**ECT主机，AS系统**能够代表**ECI客户端**执行各类函数。这些函数构成[ITU-T J.1012]中高级安全API的基础。一个“事件”将异步发生事件报告回**ECI客户端**。可能无响应。所有其他函数被指定为**ECI客户端**发起的异步或同步信息；它们的返回值表明响应状态。表8-13列出了这些函数：

表8-13 – 高级安全函数概述

函数名称	描述	条款
reqAsInitSlot	初始化一个AS槽	8.2.4.2
reqAsAStartDecryptSession	在一个AS槽内启动一个解密会话	8.2.4.3
reqAsCoupleDecryptSession	将两个解密会话耦合成一个	8.2.4.3
reqAsDecoupleDecryptSession	解耦两个耦合的解密会话	8.2.4.3
reqAsStartEncryptSession	启动一个加密会话	8.2.4.3
callAsNextKeySession	换成下一个随机密钥	8.2.4.3
reqAsStopSession	停止一个会话	8.2.4.3
reqAsExportConnSetup	设置一个从解密到加密会话的导出连接	8.2.4.4
reqAsExportConnEnd	终止现有的导出会话	8.2.4.4
reqAsLoadLk1	在密钥阶梯中为一个会话装载顶层链接密钥	8.2.4.5
reqAsComputeEncrCw	计算加密控制字	8.2.4.6
reqAsComputeDecrCw	计算解密控制字	8.2.4.7
reqAsComputeAkClient	计算ECI客户端使用的鉴权密钥	8.2.4.8
reqAsClientChalResp	代表ECI客户端使用鉴权密钥	8.2.4.8
reqAsAuthDecrConfig	采用鉴权机制（解密模式）鉴权会话配置	8.2.4.9
reqAsAuthEncrConfig	采用鉴权机制（加密模式）鉴权会话配置和加密参数	8.2.4.9
reqAsLdUssk	装载微服务器密钥	8.2.4.10
reqAsMInikLk1	计算异步微客户端初始化信息	8.2.4.11
reqAsClientImageDecrKey	计算ECI客户端图像的解密密钥	8.2.4.12
getAsSlotRk	读取槽随机密钥	8.2.4.13
getAsSessionRk	读取会话随机密钥	8.2.4.13
getAsSessionLimitCounter	读取会话随机密钥的剩余单元	8.2.4.13
setAsSessionLimitEvent	达到剩余单元发送事件的限值	8.2.4.13
reqAsEventSessionLimit	达到限值的事件信息	8.2.4.13
getAsClientRnd	为ECI客户端应用程序获取一个新的随机数	8.2.4.13
getAsSC	在一个会话中获取内容的当前加扰控制字段状态	9.9
reqAsEventCpChange	关于一个加密会话的输入内容中内容属性变更的事件信息	9.9
setAsPermitCPChange	在一个加密会话中启用/禁用在用于加密的控制字选择时生效的导入内容属性CP变更	9.9
setAsSC	设定一个加密会话的加密内容的加扰控制字段	9.9
reqAsEventSC	关于会话中加扰控制字段的变更的事件信息	9.9

本条的子条款中的伪码包含作为函数的返回值的差错代码。第8.2.4.15条对差错代码值进行了定义，包括文字描述。

8.2.4.2 AS槽初始化

在装载时间内**ECI主机**应在**高级安全系统**中代表每个待装载的**ECI客户端**保留一个**AS槽**。**ECI主机**调用如下定义的reqAsInitSlot函数。所有**AS槽**的状态信息应被设定为缺省状态；任何**导出连接**应被重置。**ECI主机**应使用装载程序核心（见第11条）装载**ECI客户端**。**AS槽**的*POClRLVnr*应反映用于确认客户端图像的**POC撤销清单**的最低版本号。该值将在**ECI客户端**发起一个会话时被验证。

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

语义：

所有**AS槽slotId**状态信息应被设定为缺省状态；任何**导出连接**应被重置。

POPK的装载需要为CPS系统提供处理链。POPL链的处理规定在第10.4条中作了规定。ECI_Certificate_Chain在[ITU-T J.1012]第5.4.1条进行了规定。一旦成功确认，应执行以下c-代码：

```
/* initialise the slot state */
ss[slotId].popk = /* validated value of popk returned by CPS */;
```

```
ss[slotId].POClRLVnr = /* value used for client image verification */;
ss[slotId].version = slotVersion;
ss[slotId].slotMode = slotMode;
ss[slotId].configAuthMode = ConfigAuthModeNone;
ss[slotId].rkSlot = rnd128();
return ErrOk;
```

第A.3条中定义的rnd128()函数返回一个随机128比特数作为16位uchar的一个阵列。

8.2.4.3 AS槽会话和随机密钥控制

AS槽支持不同并发会话的不同会话状态。以下函数为一个槽启动和停止会话：

```
int reqAsAStartDecryptSession(uintslotId, ushortmh, PubKeyspk,
SessionConfigconfig, uint*sessionId)
```

语义：

执行以下c-代码：

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

/* check if a valid client revocation list was used */
if (config.decryptConfig.clientVersion>
ss[slotId].clientPOClRLVnr) return ErrRevocEnforce;

/* locate any free sessionId;any algorithm is ok */
int i=0;
while (i<NSESSIONS && ss[slotId].se[i].active) i++;
if (i==NSESSIONS) return ErrNoMoreSessions;
/* i contains a non-active session administration block */
*sessionId = i;

/* initialise session state */
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;
```

先决条件：

- **AS槽**被成功初始化。

注 – mh（媒体处理）参数允许**ECI主机**识别与它启动的内容解密会话相关的AS解密会话。**AS系统**本身并不使用。

提供coupleDecrypSessions函数用于耦合两个初始化的会话。第二个会话被耦合至第一个会话中；第一个会话成为合并的内容的主柄。

```
int reqAsCoupleDecryptSession(uintslotId, uintsId1, uintsId2)
```

语义：

执行以下c-代码：

```
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;
/* the Secure Video Path is informed on the session coupling &/

return ErrOk;
```

先决条件：

- 两个**AS槽**会话都被成功初始化。

可调用以下函数解耦一个耦合的会话：

```
int reqAsDecoupleDecryptSession(uintslotId, uintsessionId)
```

语义：

执行以下c-代码：

```
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;
/* the Secure Video Path is informed on the session decoupling */

return ErrOk;
```

先决条件：

会话先前被耦合。

初始化一个加密会话的函数为：

```
int reqAsStartEncryptSession(uintslotId, ushortmh, uintimportSlotId,
int importSessionId, PubKeyspk, SessionConfigconfig,
uintnEncr, PubKeyencrSpk[MaxSpkEncr],
PubKeyencrPopk[MaxSpkEncr], ulongencrCwUri, uint*sessionId)
```

语义：

执行以下c-代码：

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

/* locate free sessionId;any algorithm is ok */
int i=0;
while (i<NSESSIONS && ss[slotId].se[i].active) i++;
if (i==NSESSIONS) return ErrNoMoreSessions;
/* i contains a non-active session administration block */

/* check if a valid client revocation list was used */
if (config.encryptConfig.microServerVersion>
ss[slotId].clientPOClRLVnr) return ErrRevocEnforce;

*sessionId = i;

/* initialise session state information */
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];
}

/* initialise random key state */
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);

/* initialise import state */
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;
```

先决条件：

- **AS槽**被成功初始化。

ECI主机可以采用以下函数将一个会话的随机密钥状态向前移动（将下一个移至当前）：

```
int callAsNextKeySession(uintslotId, uintsessionId)
```

语义：

执行以下c-代码：

```
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;
```

先决条件：

- **AS槽**会话被成功初始化。

当工作于TS模式时，**安全视频路径**将向相关**ECI客户端**发出当前/下一个控制字的转换的信号（见[ITU-T J.1012]）。**ECI客户端**可使用这一消息来触发对下一个控制字的计算。

ECI主机可以停止会话，作为结果，使用以下函数来终止任何来自该会话的挂起的**导出连接**：

```
int reqAsStopSession(uintslotId, uintsessionId)
```

语义：

执行以下c-代码：

```
int i, j;

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

/* decouple from any coupled decryption sessions */
for (j=0; j<NSESSESIONS; j++)
    if (ss[slotId].se[j].coupledSessionId == sessionId)
        ss[slotId].se[j].coupledSessionId = -1;
        /* the Secure Video Path is informed of decoupling */

/* cancel all export sessions */
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;
```

先决条件：

- **AS槽**会话被成功初始化。

8.2.4.4 AS槽导出控制

导出**鉴权机制**允许**ECI主机**创建一个从解密**ECI客户端**的**AS槽会话**到一个**微服务器**的**AS槽**会话的**导出连接**，从而允许从解密**ECI客户端**到**微服务器**的内容转移。**AS系统**采用**鉴权处理子系统**，使用导出**AS槽**会话POPK和minClientVersion作为确认导出和后续的导入链的基础来处理所要求的导出、导入和导出鉴权链接。最终结果是一个导出组ID的**导出连接**元被肯定确认，或连接被拒绝。从一个（导出）会话到一个导入会话创建实际连接。

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

语义：

expCh是从POPK到TPEGC或ESC**证书**的导出链。ImpCh是从TPEGC到ESC的导出链。

注 – impCh可以为空。auth[]是需要对导入链的字段共同鉴权的导出鉴权链的序列。

[ITU-T J.1012]第9.5.2.4.2条定义了CertSerialChain结构定义。

AS槽首先用auth[]导出鉴权链和安装的**ECI根密钥**和**撤销清单**版本号码验证impCh。

之后**AS槽**要求**证书处理子系统**使用POPK处理导出加导入链，AS状态寄存器**POPK**和**ExportRlVersion**为根。导出链上第一个**证书**的id应被储存在expGrpId。

一旦鉴权成功，一个导出元被添加到**AS槽**会话状态，包含导出组id和槽id加上鉴权的导出**ECI客户端**的会话id。应执行以下c-代码以继续创建导入连接。可以为两个导出组id计算鉴权，从而允许内容属性中从一个到下一个导出组的无缝转换。

```
/* the CPS delivers the following variables on successful processing of the
   Export import chains */
PubKey impSpk;           /* the spk of the importing system */
uint   impConfigVersion; /* the config. Version nr of the export system */
uint   expGrpId;         /* the export group for which the export connection     is valid */


/* check if potential import slot is in decent state */
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;
}

/* check if another import connection already exists */
if (ss[impSlotId].se[impSessId].ies.importSlotId != ImportNone)
  return ErrExportOngoing;
/* Set the import/export state of the import-session to reflect the export connection */
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;
```

先决条件：

- **AS槽**会话被成功初始化。

在设定**导出连接**后，导入方亦能够终止连接（这将实际上停止加密会话）：

```
int reqAsExportConnEnd(uintslotId, uintsessionId)
```

语义：

执行以下c-代码：

```
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;
```

先决条件：

- 建立用于导入的AS槽会话。

8.2.4.5 LK1密钥阶梯初始化

为了在**AS槽**中执行**密钥阶梯**机制操作，**ECI主机**可为后续的**密钥阶梯**输出计算装载顶层链接密钥LK1。

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

语义：

执行以下c-代码：

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

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

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

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

先决条件：

- AS槽会话被初始化。

8.2.4.6 加密控制字计算

一旦设定**AS槽**状态字段lk1，即可计算控制字。cwIndx表明被计算的是奇数或偶数控制字。该值可以是0（偶数）或1（奇数），基于文件的解密应始终为0。

```
int reqAsComputeEncrCw(uintslotId, uintsessId, ulongcwUri, uintnElk,
    SymKeyelk[24], ucharXT[32], uintrkIndx, Field2field2,
    uintcwIndx)
```

语义：

执行以下c-代码：

```
PubKeyspk[MaxSpkEncr+1], popk[MaxSpkEncr+1];/* temporary variables */
SessionConfigconfig[MaxSpkEncr+1];/* temporary variable */

/* basic consistency checks */
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;
}

/* verify if the slot configuration has been authenticated */
if (ss[slotId].se[sessId].configAuthMode != ConfigAuthModeAk1)
    return ErrNoConfigAuth;

/* verify if the CPS ECI Host Root state is sufficient to proceed */
if (!cpsEciRootStateOk(slotId,sessId)) return ErrRevocEnforce;

/* check if random slot-session key has to be applied */
SymKey rkAppl; /* random key that may have to be applied */
if (rkIndx == 0) {
        rkAppl = ss[slotId].se[sessId].rkState.rkCurrent;
} else if (rkIndx == 1) {
        rkAppl = ss[slotId].se[sessId].rkState.rkNext;
} else {
        return ErrParam7;
}

/* insert random slot key and random session key if required */
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;
}

/* compute input-C, insert in key ladder */
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]);

/* use ARK with value 0 */
uchar ark[16] = (uchar){0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/* define spk, popk and config inputs to key ladder; using slot's spk/popk in position 0 and a
replication of the slot configuration */

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 spkUri values */
ulong spkUri = (0x1<<(nSpk+1)) - 1; /* all SPKs can be used for decoding
```

```
 keys */

/* perform the key ladder calculation */
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 is sent to the encryption resource along with cwUri, msField1, cpMask and cwIndx */

return ErrOk;
```

先决条件：

- 会话的LK1被装载。
- 如有需要，**AS槽**会话被鉴权。

8.2.4.7 解密控制字计算

一旦设定**AS槽**状态字段lk1，即可计算控制字。cwIndx表明被计算的是奇数或偶数控制字。该值可以是0（偶数）或1（奇数），基于文件的解密应始终为0。

```
int reqAsComputeDecrCw(uintslotId, sessionId, ulongcwUri, uintnSpk,
    uintnElk, SymKeyelk[24], PubKeyspk[16], PubKeypopk[16], SSConfigconfig[16], uchar  XT[32],
uintrkIndx, Field2field2, uintcwIndx)
```

语义：

执行以下c-代码：

```
/* basic consistency checks */
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 ;

/* verify if the slot configuration has been authenticated if so required */
if ( ss[slotId].se[sessionId].config.decryptConfig.akModeAuth &&
     ss[slotId].se[sessionId].configAuthMode != ConfigAuthModeAk1
   ) return ErrNoConfigAuth;

/* verify if the CPS ECI Host Root state is sufficient to proceed */
if (!cpsEciRootStateOk(slotId,sessionId)) return ErrRevocEnforce ;

/* ensure proper slot spk, popk and slotConfig are applied */
spk[si]= ss[slotId].se[sessionId].spk;
popk[si] = ss[slotId].se[sessionId].popk;

/* only authenticate the slot's decrypt configuration if required */
if ( ss[slotId].se[sessionId].config.decryptConfig.klModeAuth )
    ssConfig[si].decryptConfig = ss[slotId].ssConfig.decryptConfig;

/* in all cases authenticate the klModeAuth and akModeAuth fields */
config[si].decryptConfig.klModeAuth=
     ss[slotId].se[sessionId].config.decryptConfig.klModeAuth;
config[si].decryptConfig.akModeAuth =
     ss[slotId].se[sessionId].config.decryptConfig.akModeAuth;

/* check if random slot-session key may have to be applied */
SymKey rpAppl; /* random key that may have to be applied */
if (rkIndx == 0) {
        rpAppl = ss[slotId].se[sessionId].rkState.rkCurrent;
} else if (rkIndx == 1) {
        rpAppl = ss[slotId].se[sessionId].rkState.rkNext;
} else {
        return ErrParam11;
}

/* insert random slot key and random session key if required */
```

```
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;
}

/* compute input-C, i.e. elk[nElk-1] for content Property authentication */
/* veryfy basicUri control bit is set */
if (((elk[nElk-1][0]>>2)&0b1) != 0b1) return ErrBasicUriCtrl;
uchar result1[16];
computeField1Decrypt(elk[nElk-2],result1);
computeInputC(result1, field2, elk[nElk-2]);

/* use ARK with value 0 */
uchar ark[16] = (uchar){0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

/* perform the key ladder calculation */
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 is passed to the decryption resource session along with cwUri, result1 and cwIndx and the
sessions states media handle value */

return ErrOk;
```

先决条件：

- 会话的LK1被装载。
- 如有需要，**AS槽**会话被鉴权。

8.2.4.8 计算akClient及其应用

密钥阶梯块的**鉴权机制**允许**ECI客户端**使用**鉴权机制**对安全密钥进行安全计算：

```
int reqAsComputeAkClient(uintslotId, InputVinputV, uintnSpk,
    ucharspkIndx, PubKeyspk[16], PubKeypopk[16], SessionConfigakCnf[16],
    ulongspkUri,ucharXT[32], bool online)
```

语义：

执行以下c-代码：

```
/* basic consistency checks */
if (ss[slotId].slotMode==SlotModeEncr) spkIndx = 0;
if (spkIndx>= 16) return ErrParam4;
/* check if spkUri in 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;
}
/* ensure proper slot spk and popk are applied */
popk[spkIndx] = ss[slotId].popk;

/* ensure proper ACF and ARK are applied */
ucharark[16] ;
```

```
ucharacf[15] = acfAk1Mode ;
acf[1] = AkUseCl;
if (online) {
acf[1] += AkOnline;
ark = ss[slotId].slotRk;
} else {
    acf[1] += AlOffline;
    ark = {0} ;
}

/* perform the authentication mechanism */
ss[slotId].akClient =
AuthMech(inputV,acf,ark,popk,akCnf,XT,spkUri,nSpk,spkIndx,spk);
return ErrOk;
```

先决条件：

- 槽已被初始化。

为采用计算的**ECI客户端**AK密钥定义了以下函数：

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

语义：

执行以下c-代码：

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

先决条件：

- 槽被初始化。
- 槽的AkClient已被成功计算。

8.2.4.9 AS槽会话配置鉴权

密钥阶梯块的**鉴权机制**允许**提供服务器**对该槽的会话配置鉴权。**提供服务器**可发出离线鉴权信息或通过在ACF中设置AkOnline来要求在线鉴权。两个函数分别用于对一个解密和一个加密槽的鉴权。

```
int reqAsAuthDecrConfig(uintslotId, uintsessId, InputVinputV,
uintnSpk, ucharspkIndx, PubKeyspk[16], PubKeypopk[16], SSCnfgclCnf[16],
ulongspkUri, ucharXT[32], bool online, uchar verifier[16])
```

语义：

执行以下c-代码：

```
/* basic consistency checks */
if (!ss[slotId].se[sessionId].active) return ErrParam2;
if (ss[slotId].slotMode!=SlotModeDecr) return ErrSlotMode;
if (spkIndx>= 16) return ErrParam5;
/* check if spkUri in 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;

/* verify if the CPS ECI Host Root state is sufficient to proceed */
if (!cpsEciRootStateOk(slotId)) return ErrRevocEnforce;

/* ensure proper slot spk, popk and config are applied */
popk[spkIndx] = ss[slotId].popk;
spk[spkIndx] = ss[slotId].se[sessId].spk;
clCnf[spkIndx] = ss[slotId].se[sessId].config;

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

```
}

/* perform the authentication mechanism */
Secret SymKeyak =
AuthMech(inputV,acf,ark,popk,clCnf,XT,spkUri,nSpk,spkIndx,spk);

ucharresponse[16] = AuthMechResponse(ak, verifier);

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

先决条件：

- **AS槽**会话的LK1被装载。

加密的鉴权包括验证加密特定状态。

```
int reqAsAuthEncrConfig(uintslotId, uintsessId, InputVinputV,
ucharXT[32], bool online,uchar verifier[16])
```

语义：

执行以下c-代码：

```
PubKey spk[MaxSpkEncr+1], popk[MaxSpkEncr+1]; /* temporary variables */
SessionConfig config[MaxSpkEncr+1]; /* temporary variable */
/* basic consistency checks */
if ((ss[slotId.SlotMode != SlotModeEncr) return ErrSlotMode;
/* verify if the CPS ECI Host Root state is sufficient to proceed */
if (!cpsEciRootStateOk(slotId,sessId)) return ErrRevocEnforce;
/* define spk, popk and config inputs to key ladder; using slot's spk/popk in position 0 and a
replication of the slot configuration */
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 spkUri values */
ulong spkUri = (0x1<<(nSpk)) - 1;
                        /* all SPKs can be used for decoding content */
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};
}
/* perform the authentication mechanism */
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;
}
```

先决条件：

- **AS槽**会话的LK1被装载。

8.2.4.10 装载一个微服务器密钥

微服务器客户端可使用在异步服务器模式中操作的**AS槽**并装载一个**微服务器**密钥值ussk用于后续创建一个到**微客户端**芯片组的安全连接：

```
int reqAsLdUssk(uintslotId, uintsessId, InputVinputV,
ucharXT[32], bool online, ucharmUssk[NUSSK])
```

语义：

执行以下c-代码：

The following C-code shall be executed:

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

/* basic consistency checks */
if (ss[slotId].slotMode!=SlotModeEncr) return ErrSlotMode;
if (!ss[slotId].se[sessId].config.encryptConfig.asymKlMode)
        return ErrSlotModeUndefined;

/* verify if the CPS ECI Host Root state is sufficient to proceed */
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 spkUri values */
ulong spkUri = (0x1<<(nSpk+1)) - 1; /* all SPKs can be used for decoding
 keys */

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};
}

/* perform the authentication mechanism */
Secret SymKey ak =
      AuthMech(inputV,acf,ark,popk,config,XT,spkUri,nSpk,0,spk);

/* perform AES ECB decoding of 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;
```

先决条件：

- 会话配置被鉴权。

8.2.4.11 为微客户端生成MinitLk1

在异步**微处理器**模式中，**AS槽**可为**微客户端**生成**密钥阶梯块**初始化信息：

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

语义：

[ITU-T J.1012]第5.4.1条定义了ECI_Certificate_Chain，并包含用于验证**微客户端**的**证书链**。该函数首先使用CPS来确认使用槽[slotId]并以POPK作为**父证书**、ss[SlotId].se[sessId].config.encryptConfig.microServerVersion作为链中第一个**证书**的最低**撤销清单**版本号的ClCPK。若验证成功，变量clcpk包含客户端芯片组公钥，并执行以下c-代码：

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

先决条件：

- Ussk被初始化。
- 会话处于异步加密模式。

8.2.4.12 计算ECI客户端图像解密密钥

为执行加密图像装载，**AS槽**可提供一个鉴权密钥，可使用该鉴权密钥来解密解码图像的密钥。必须在槽初始化之前执行这一函数：

```
int reqAsComputeImageKey(uintslotId, InputVinputV,
symKeyeKey,bool online, ECIRootStatemin_root_state)
```

语义：

执行以下c-代码：

```
/* a default slot configuration state is used */
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;

/* create straightforward popk/spk, XT, clCnf, */
PubKey popkArr[1]; /* also used for 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} ;
}
```

```
/* perform the authentication mechanism */
Secret SymKey ak =
      AuthMech(inputV,acf,ark,popkArr,cnf,XT,spkUri,1,0,popkArr);

Secret SymKey dImgKey = AuthMechResponse(ak, eImgKey);
/* dImgKey is subsequently used by the client loader to decrypt the client image using AES CBC mode
with IV=0 */

return ErrOk;
```

先决条件：

- 槽被设置为缺省状态；slotRk被设置为新的随机值。

注 – 该函数并非在**ECI客户端**的要求下执行。

8.2.4.13 读取高级安全信息

AS系统为**ECI客户端**提供对其生成数据的访问，并为**ECI客户端**提供通用随机密钥函数。

注 – 本条定义的“获取”和“设置”函数并不会在未定义参数值上生成自动误差，只是以防“获取”函数仅返回一个未定义的值，“设置”函数未生效。

以下函数读取**AS槽**的随机密钥（通常用作会话的当前状态）：

```
SymKeygetAsSlotRk(uintslotId)
```

语义：

执行以下c-代码：

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

若槽未初始化，会返回一个数字。

以下函数读取会话的随机密钥状态：

```
SymKeygetAsSessionRk(uintslotId, uintsessionId, uintrkIndx)
```

语义：

执行以下c-代码：

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

若槽未初始化，会返回一个数字。

会话的随机密钥的限值计数器可被读取：

```
ulonggetAsSessionLimitCounter (uint slotId, uint sessionId)
```

语义：

执行以下c-代码：

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

若槽未初始化，会返回一个数字。

限值计数器值可以被设置为到达该数值则引发一个事件（即，及时充分地更新随机密钥）：

```
ulongsetAsSessionLimitEvent(uintslotId, uintsessionId, ulongeventLimit)
```

语义：

调用这一函数之后，一旦以下条件为真则引发eventSessionLimitCounter事件：

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

注2 – 第二个调用覆盖前一个调用。用一个非常大的值第二次调用这一函数用于eventLimit则立即取消该事件（除非该事件已经被引发）。

达到会话的事件限值时引发下列事件：

```
reqAsEventSessionLimit(uintslotId, uintsessionId)
```

注3 – 在[ITU-T J.1012]中这一事件转化为无对应响应的异步信息。

8.2.4.14 生成客户端随机数

ECI客户端可通过调用以下函数来请求一个由**AS系统**生成的128比特随机数。

```
SymKeygetAsClientRnd()
```

语义：

执行以下c-代码：

```
return rnd128();
```

8.2.4.15 错误代码

表8-14定义了第8.2.4条定义的函数返回的错误代码。

这些错误代码遵循[ITU-T J.1012]第9条定义的**ECI主机**和**ECI客户端**之间的信息的错误代码惯例。

表8-14 – 错误返回代码定义

错误返回代码	值	描述
ErrSlotMode	-256	AS槽未处于适用这一操作的模式
ErrNoMoreSessions	-257	没有更多会话可用
ErrSession1Coupled	-258	首个会话已被耦合
ErrSession2Coupled	-259	第二个会话已被耦合
ErrSessionNotCoupled	-260	会话未被耦合
ErrNoSuchSession	-261	会话不存在
ErrExportNoSlot	-262	导出槽未知
ErrExportSlotBadState	-263	导出槽处于不当状态
ErrExportOngoing	-264	导出会话已有一个导出连接
ErrImportSlotBadState	-265	导入槽不处于加密模式
ErrNoExport	-266	当前没有至会话的导出
ErrSpkUriViolation	-267	槽SPK的SpkUri对于槽模式有不正常值
ErrSlotModeUndefined	-268	槽模式对于本次操作有不正常值
ErrRevocEnforce	-269	ECI撤销不允许槽进行操作
ErrNoConfigAuth	-270	槽配置未被适当鉴权
ErrNoSlotRkInsert	-271	ELK矢量对于插入随机密钥来说不够长
ErrSpk0NoDecrypt	-272	spk[0]无法被用于生成解密控制字
ErrBasicUriCtrl	-273	基本URI field1控制比特未设定
ErrOk	0	成功调用
ErrSlotConfigAuthFail	-274	槽会话配置的鉴权失败
ErrParam<N>	-<N>	输入参数N错误（ErrParam1值为-1并在参数1中标明错误）
	1..MaxInt	成功调用，值由信息定义来定义。

9 加扰/解扰和内容导出

9.1 基本功能性

安全视频路径可解密内容。该内容带有**内容属性**和**导出连接**。如果**内容属性**允许的话，内容可以被转发到标准输出，在匹配**导出连接**的情况下可以被**微服务器**重新加密。

出于信息管理的目的，**ECI**定义了解密和加密资源。资源被用于每次从一个带有一个单独CW的加密或待加密的媒体会话解密或加密内容，解密或加密资源被连接至一个单独的解密或加密**AS**槽。对于TS流解密来说，解密资源对一个奇数和一个偶数CW有一个双缓存。奇数或偶数CW由待被解码的流选择。这可以满足在待加密的内容的**内容属性**发生变化的情况下飞速更改控制字的需要。对于基于文件的解密和加密来说，**ECI主机**提供CW和待解密的内容之间的同步，这比实时快得多。基于文件的解密和重新加密资源只需要一个单独的CW缓存。

注：需要两个或更多控制字来对不同基本数据流解扰的TS流需要多个解扰源，从而需要多个会话。

ECI不指定任何关于缓冲或（可能广泛的）中间处理（例如可在解密内容从解密传递到加密资源的解密内容上执行的编码转换或加水印）的特性。这类处理可导致巨大延迟。CPE制造商可选择造成解密资源和相关的重新加密资源之间的时间偏移量的适当操作。重新加密槽和**ECI客户端**与内容的加密同步。

9.2 加扰器和解扰器规范

一个**ECI** CPE的解扰函数应支持以下TS模式的解扰算法：

- CSA1/2，PES和TS模式均在[ETSI ETR 289]和[b-ETSI DVB CSA]中定义。
- CSA3，PES和TS模式[ETSI TS 100 289]和[b-ETSI DVB CSA3]。
- DVB-CISSA PES和TS模式[ETSI TS 103 127]。

ECI CPE的解扰函数应支持文档模式的以下解扰算法：

- 在[ISO/IEC 23001-7]中定义的CENC AES128 CTR模式和AES128-CBC模式（均为全样本和子样本加密）。MPEG-DASH的CENC和[ISO/IEC 23009-4]。

ECI CPE的加扰函数应支持以下TS模式下的加扰算法：

- DVB-CISSA PES和TS模式[ETSI TS 103 127]。

ECI CPE的加扰函数应支持以下文档模式的加扰算法：

- [ISO/IEC 23001-7]和[ISO/IEC 23009-4]中定义的CENC AES128 CTR和CBC模式 （均为全样本和子样本加密）。安全视频路径应为加密的内容生成一个独一无二的初始化矢量，附有用于AES-CTR模式的单个CW，并遵循[ISO/IEC 23009-4]中定义的AES-CBC模式IV定义规定。ECI主机可访问初始化矢量用于包内容。

9.3 导出控制

被鉴权的**导出连接**被解密**AS槽**会话用作授权解密资源进行导入和导出的权证。如果相关**AS槽**会话提供的**导出连接**允许一个导出组ID这样做的话，解密资源应允许解密内容导出至加密资源，并且解密内容信号的**内容属性**表示第8.2.4.4条定义的对应导出组ID。如果由**内容属性**中的导出组ID选择的导出组的**导出连接**并非由相关**AS槽**提供的已被验证的**导出连接**，则解密资源不得允许解密内容被导出至一个加密资源。

9.4 输出控制

输出控制**内容属性**被用于禁用或启用处于对CPE的输出连接行业标准保护技术的保护下的内容导出。如果相关**AS槽**会话的输出控制信息允许的话，解密资源应允许解密内容被导出至一个输出端。如果相关**AS槽**会话的输出控制信息不允许的话，解密资源不得允许解密内容被导出至输出端。

9.5 耦合会话上的内容属性比较

安全视频路径应验证会话的field1中定义的**内容属性**（头两个字节除外）是否等于任何耦合的会话的**内容属性**。允许具有同等**内容属性**的耦合的会话的内容导出和输出。从**ECI**保护的角度来看，从那时起合并流应被视为一个会话。如果field1的**内容属性**（头两个字节除外）不相等，则耦合流的合并被禁止。

9.6 导出时的内容属性传播

解密资源会话应传播由客户端设置并（部分）由**密钥阶梯**隐性鉴权的field1**内容属性**以及内容至第8.2.4.6条定义的导入解密内容的重新加密会话资源。接收解密内容的加密会话对照为field1设定的用于加密内容的值检查指定的field1字节，同时采用一个掩膜来选择由函数定义的需要传播的字段，从而确保指定的解密客户端field1字节被传播至加密内容。

加密资源会话在每个输入值impField1、expField1和cpMask变更时执行以下c-代码：

```
uchar impField1[16];/* field1 values for the imported content */
uchar expField1[16];/* field1 values from the encryption CW computation */
ushortcpMask;/* comparison mask */

bool propOk = true;/* indicates if propagation of imported content is Ok */
int i;

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

if (propOk) /* re-encrypt content */
else /* do not re-encrypt content */
```

9.7 导出时的基本URI执行

从解密**AS槽**会话到重新加密**AS槽**会话的基本URI传播通过以下机制来控制，其中包含加密槽的IDslotId和加密会话的IDsessionId：

由加密**AS槽**指配给用于基本URI的内容的权利并不比与传播内容相关的权利更加自由。

微服务器被鉴权：ss[slotId].se[sessionId].config.decryptConfig,akModeAuth等于0b1。

若基本URI不允许内容重播（即流模式），在导出时应检查以下内容：

– ss[slotId].se[sessionId].config.decryptConfig.rkDecrMode.mode模式应与RKModeNone不相等（即，在系统重启时采用随机当前状态防止之前编码的内容的重播）；并且

– ss[slotId].se[sessionId].klModeAuth 应被设置（值 0b1）为确保服务器使用的decryptConfig（包括**微客户端**上的随机密钥插入）得到鉴权，并由**微客户端**基于**密钥阶梯**计算来使用。

9.8 行业标准输出上的内容属性应用

标准输出（通常为与行业标准保护系统结合的物理输出）应使用**内容属性**来选择合适的输出保护设置或在没有适当设定的情况下禁用输出。合规规定中对此进行了准确规定定义。

基本URI和输出控制**内容属性**执行的**稳健性**应与**安全视频路径**处于相似水平。

标准URI执行的**稳健性**应至少与**ECI主机**执行的**稳健性**一样高，但对有复杂执行要求的函数可有适当例外。

9.9 控制字同步

为处理TS流，**安全视频路径**提供以下允许对控制字变更（用于加密）进行控制的函数，并在加扰控制字段变更时提供通知。这一会话中的函数和事件遵守第8.2.4条定义的惯例。

AS 槽会话可为内容加密或解密提供“奇”和“偶”两种控制字。

在解密情况下，加扰控制字段[ETSI TS 100 289]通知解密函数使用哪个控制字。当内容被表示为未被加扰时，不使用任何控制字。结果的值等于加扰控制字段，[ETSI TS 100 289]第5.1条定义了值。

以下函数读取流中的加扰控制字段的当前状态：

```
uintgetAsSC(uintslotId, uintsessionId)
```

在加密的情况下，应用的控制字可基于两个事件变更：

1) 导入内容的**内容属性**的变更会触发用于加密的控制字的变更。**AS槽**可推迟这一变更，以完成由后继事件触发的控制字中正在进行的变更。

   AS槽会话发出应用的控制字必须变更的信号。

在导入内容非加扰的情况下，加密不采用加扰，应在首个可能变更的位置将加扰控制字段设置为0b00。反之亦然：在输入内容从非加扰转变为加扰的情况下，应使用下一个控制字对内容进行加扰；在与内容的清楚段之前加扰的内容相比较时，选择相反的密钥。

表示导入内容属性变更的事件定义为：

```
reqAsEventCpChange(uintslotId, uintsessionId)
```

语义：

事件表示导入内容的**内容属性**的变化（如果此类内容需要加密的话）。

安全视频路径不得允许加密参数和导入的**内容属性**之间存在差异的时间段长于[b-ITU-T J. Suppl. 7]第6.6.2条中建议的最大值。

注1 – 从加密到非加密的导入内容的变更时不会引发这一事件。**内容属性**不适用于非加密内容。

在收到如下指令时，**安全视频路径**允许**AS系统**延迟任何至**内容属性**的eventCpChange自动变更：

```
setAsPermitCPChange(uintslotId, uintsessionId, bool permit)
```

语义：

本函数设置许可，以允许导入内容的控制属性中的自动变更触发加密内容的控制字内的变更。

注2 – 这一函数应高于任何未基于即将到来的**内容属性**计算的下一个控制字，即只反映当前状态或随机密钥变更。

注3 – 若变更许可被禁用(permit==false)，应在允许的导入内容的内容属性不一致的时间段内恢复，从而不会导致重新加密流的“中断”。

以下函数允许将加密的加扰控制字段设置至某一状态。

```
setAsSC(uintslotId, uintsessionId, uintscramblingControlField)
```

语义：

加扰控制字段的值被设定为流中的第一个可能的变更点上的加扰控制字段的值。加扰控制字段仅允许0b10和0b11两个值（分别为偶数和奇数密钥的加扰）。

在导入内容包含一个非加密状态的情况下，加密流的加扰控制字段被设定为0b00（无加扰）。

为解密和加密会话定下以下事件函数：

```
reqAsEventSC(uintslotId, uintsessionId, uintscramblingControlField)
```

语义：

该事件在加扰控制字段状态变更时引发。

10 证书处理子系统

10.1 证书链基本处理规定

证书处理子系统可处理**证书链**以对项目鉴权，基于初始公钥和最小**撤销清单**号。大部分证书链处理是通用的。本条定义了**证书链**的通用处理规定。以下各条定义了对于不同类型链的具体处理规定。

[ITU-T J.1012]第5.4条中定义了以下**证书链**。

CPS规定定义采用循序渐进的方式，利用初始公钥和最小**撤销清单**号来处理在链起始处启动的**证书链**（第一个**撤销清单**）。第一步是验证**撤销清单**。第二步验证链上的下一个**证书**。一旦完成第1和第2步之后，将定义新的公钥和**撤销清单**号，以处理链的剩余部分。重复第1和第2步，直到处理完整个链。一般来说，建议提供链的软件函数预先对这些链进行验证，以防止出现CPS意外地无法处理该链的情况。

通常一个**证书链**的处理步骤为：

1) CPS应在**撤销清单**上执行以下验证的验证：
   a) CPS应检查撤销清单format_version字段，以匹配其可解译的版本（见链的具体处理规定）以及rl_id.type和rl_id.rl_indicator字段以匹配预期值。
   b) 若父版本为根证书（root_version_indicator=1），则ECI主机应选择root_version为父的根证书，否则将使用预先装载或先前的证书。
   c) CPS应采用最后被验证的公钥来验证撤销清单的签名。
   d) CPS应验证撤销清单的长度是否对应其字段值，以及任何可变长度字段的长度适宜。
   e) CPS应验证最小撤销清单数是否未使撤销清单的版本号失效。
2) CPS应在**证书**上执行以下验证：
   a) CPS应验证链中**证书**的**下一个**<类型，实体_id，版本>是否未根据上一个**撤销清单**被撤销并创建最低**撤销清单**版本以根据上一个**撤销清单**的**base_rl_version**和**min_rl_version**字段来配合该**证书**。
   b) CPS应检查**撤销清单format_version**字段以便与其允许解译的版本相匹配。
   c) CPS应检查**证书**的长度是否与其字段值相对应，以及任何可变长长度字段的长度适宜。
   d) CPS应使用公钥验证**证书**的签名。

在处理步骤1和2之后，公钥和最小**撤销清**单被更新。公钥等于步骤2中处理的**证书**的公钥字段，在步骤2a中获得的最小**撤销清单**版本。

并非所有**证书**都需要配有撤销清单。如果一个证书certificate-id的类型字段的最重要比特等于零，则**证书处理子系统**应要求**证书**附带**撤销清单**以便进行进一步链处理。在无需**撤销清单**的情况下，不适用在上述步骤中的任何对于**撤销清单**以及版本号和**撤销清单**版本号的处理。

10.2 主机图像链特别规定

CPS适用于对主机图像链的特别验证：

1) 第一个**撤销清单**为0x1类型（制造商撤销清单）。
2) 第一个**证书**为0x1类型（制造商**证书**）。
3) 第二个**撤销清单**为0x0类型（**ECI主机撤销清单**）。
4) 第二个**证书**为0x0类型（**ECI主机证书**）。
5) 可能的第三个**证书**为0x98类型（**主机**图像序列**证书**）。

应使用上一个**证书**（**ECI主机证书**或**ECI主机**图像序列**证书**）的公钥来验证实际**ECI主机**图像。

10.3 客户端图像链特别规定

CPS应适用于对客户端图像链的特别验证：

1) 第一个**撤销清单**为0x2类型（供应商**撤销清单**）。
2) 第一个**证书**为类型0x2（供应商**证书**）。
3) 第二个**撤销清单**为0x0类型（**ECI客户端撤销清单**）。
4) 可能的第二个**证书**为0x1类型（客户端序列**证书**）。

应使用上一个**证书**（供应商**证书**或客户端序列证书）的公钥来验证实际**ECI主机**图像，在上一个**证书**是供应商**证书**的情况下虑及用于验证图像版本的客户端**撤销清单**的上一个版本号。

10.4 平台操作证书特别规定

CPS适用于对**平台操作证书链**的特别验证：

1) 第一个**撤销清单**为0x3类型（**运营商**撤销清单）。
2) 第一个**证书**为0x3类型（**运营商证书**）。
3) 第二个**撤销清单**为0x0类型（**平台操作**撤销清单）。
4) 第二个**证书**为0x0类型（**平台操作证书**）。

10.5 导出/导入链特别规定

10.5.1 导出鉴权链处理

应向CPS提供导出鉴权链和第三方导出链对应部分。

CPS应从ss[slotId].se[sessionId].config.decryptConfig.minEciRootState中定义的最低根版本和**撤销清单**版本开始。应处理EAOC链和EAC**证书**以及验证以下针对该链的特别规定的相关**撤销清单**：

- 根RL的id为0x4（导出鉴权**运营商**撤销清单）。
- 下一个**证书**（EAOC）的id为0x4。
- 下一个**撤销清单**（REAOC RL）的id为0x0。
- 链中的下列**证书**（EAC）的id为0x0。
- **证书**的延伸字段的内容应等于导出链上的对应导出链**证书**。
- 链中的下列**撤销清单**（EAC-RL）的id为0x0。
- 导出链上的所有**证书**应被导出鉴权链循序验证。
- 第三方导出链部分的第一个**证书**应为TPEGC（证书id等于0x5）。
- 第三方导出链部分的最后一个**证书**应为TPEGC、ESC或ERC（证书id分别等于0x5、0xE、0xF）。
- 中间**证书**应为EGC（证书id等于0x4）。
- 若上一个证书为TPEGC，则它应作为下一个导出链部分的起点。导出鉴权链的所有后续部分和第三方导出链部分应重复上述验证流程，直到获得经充分验证的第三方导出链结果为止（以ESC或ERC结束）。

10.5.2 导出链验证

CPS应从POC公钥、要为之创建导出的导出组指数，以及从**AS槽**状态ss[slotId].se[sessionId]. config.decryptConfig.minClientVersion字段中获得的应施用于POC**撤销清单**的最低撤销清单版本号开始。

注：此类验证依赖于适用的PORK和**撤销清单**版本的鉴权。应使用AK模式鉴权或采用**密钥阶梯**的隐式鉴权（见第8.2.2.2条klModeAuth和akModeAuth字段）来建立。

CPS应将POC-RL、EGC和EGC-RL以及后续的TPEGC或ESC作为常规**证书链**来处理。应验证以下附加规定：

- EGC的类型为0x4。
- EGC的export_group_id字段应等于导出组指数。
- EGC撤销清单的类型为0x4。
- EGC-RL的类型为0x4。
- TPEGC或 ESC的类型与[ITU-T J.1012]表5.2-2中的值对应。

第10.5.3条规定了TPEGC的处理。第10.5.4条规定了ESC的处理。

10.5.3 第三方导出链验证

第三方导出链的处理始于对前导TPEGC及其**撤销清单**的最低**撤销清单**版本号的验证。处理以一个ESC结束。

10.5.4 导出系统证书处理

采用ESC**证书**SPK（ESC的公钥）和ESC**父**最低**撤销清单**版本号来验证**导出连接**。**证书**SPK应匹配指定导出槽的ss[slotId].spk字段。最低**撤销清单**版本号应大于导出ss[slotId].ssConfig.microServerVersion。

注： 导出槽SPK和微服务器版本应由**AS槽**的AK**鉴权机制**来鉴权，以确保鉴权有意义。

10.5.5 目标客户端链处理规定

目标客户端链处理始于PORK和**微服务器MSConfig**状态的最小**撤销清单**。由CPS执行的**目标**客户端链处理应遵循第10.1条定义的通用规定。除此之外，由CPS执行的**目标**客户端链处理应遵循以下特别规定：

1) 第一个撤销清单为0x0类型（目标撤销清单）。
2) 第一个证书为0x0（目标组证书）或0x8（微客户端证书）类型。
3) 在第2步证书为目标组证书的情况下，重复第1步和第2步。

得到的**微客户端证书**公钥是应根据第7.3条描述的机制使用的芯片组公钥。

10.6 CPS ECI根密钥初始化

在**AS系统**的初始化期间，**ECI主机**装载包含关于适用的**ECI根密钥**和**撤销清单**数的最新信息的CPS。

```
function InitCPSEciRoot(ucharminRootKeyVersion, uintminRevListNr)
```

语义：

执行以下c-代码：

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

CPS应采用**rootKeyVersion**作为**ECI根密钥**版本号，以及对所有提供给它的链使用**minRevListNr**用于装载**ECI**凭证。

AS系统的所有其他状态将被重置。

要注意的是，**ECI主机**的两个参数的设置都应确保所有**ECI客户端**能够被装载，并且**ECI主机**不被撤销，虽然并无**ECI客户端**遭到**撤销**。

11 装载程序核心

11.1 引言

ECI系统使用一个允许**ECI客户端**安全地验证装载的**ECI主机**版本和**ECI客户端**凭证从而探测任何已知的安全问题的装载程序核心。该系统允许**ECI主机**和**ECI客户端**（包括图像和POPK）作为常规系统操作函数来更新。

ECI主机和**ECI客户端**图像的装载程序依赖于以下条款中被作为规定来定义的某些**稳健性**规定。这些规定的执行的稳健性应由在**ECI**规定范围以外的合适文件来定义，但总体上这些规定将拥有同等的执行**稳健性**。一些规定被认为需要执行更高的（最高等级）**稳健性**，并且大大超过**ECI主机**的执行的稳健程度。

11.2 主机装载程序规定

ECI主机装载程序应遵守以下规定：

1) **ECI主机装载程序**应确保用于验证**ECI主机**图像的**ECI**根版本和**ECI**根**撤销清单**版本号在接通电源初始化时被存储，并且在此之后无法更改此号码。这一规定需要最高等级**稳健性**。

2) **ECI主机装载程序**本身不得被修改。这一规定需要最高等级**稳健性**。

3) 在需要防止操纵敏感信息或遵守保密信息的范围内，**ECI主机**图像一旦被装载就无法被修改或评论。

4) 由软件执行的任何源自前一个图像的后续**ECI主机**图像验证（在分段装载程序的情况下）应使用同一个**ECI主机证书**公钥和**撤销清单**用于验证。

建议分段装载程序采用亦被用于验证第一个装载的**ECI主机**图像的单独安全机制来验证E**CI主机**图像。

11.3 客户端装载程序规定

ECI客户端装载程序存在于**ECI主机**的语境中。**ECI主机**设置其在装载任何客户端相关项目之前用于验证**证书链**的最小**ECI**根版本和**ECI**根撤销清单版本。**ECI客户端装载程序**遵守以下规定：

1) ECI客户端图像应首先被解密，如果第11.5条定义做出此要求的话。

2) ECI客户端图像和PORK应采用第10条定义的CPS处理过的链来验证。这一规定需要最高等级稳健性。

3) ECI客户端图像或客户端序列图像证书（如适用）应与PORK和平台操作客户端撤销清单共同验证。这一撤销清单的版本号的充裕度随后在AS槽会话初始化时由ECI客户端验证。

4) ECI客户端图像一旦被装载就无法被修改或评论。

5) ECI客户端不得“打破其沙箱”并观测或修改ECI主机或ECI客户端行为。

11.4 撤销执行

ECI采用稳健的执行机制用于**ECI主机**和**ECI客户端**图像凭证的验证。这一机制按照以下规定操作：

1) 解扰器在用于验证ECI主机证书链的ECI根版本和最小根撤销清单版本号低于ECI主机在初始化时装载的版本和版本号时应停止操作。这一规定需要最高等级稳健性。

注1 – 由于**ECI主机**根**撤销清单**应通过所有**运营商**渠道被定期更新，并且**ECI主机装载程序**可使用最新的**ECI主机**根**撤销清单**，这一情况并非典型。

2) 按照第11.2条的规定，AS系统拒绝装载任何无法使用ECI主机在初始化时设置的ECI根版本和最小ECI根撤销清单号来验证其证书链的ECI客户端。这一规定需要最高等级稳健性。

3) 在ECI客户端所要求的最小根版本号和最小根撤销清单版本号低于ECI主机在初始化时装载的根版本号和撤销清单版本号时，AS槽拒绝计算公钥。在第8.2.4条中定义的客户端图像、加密和解密密钥的计算规定中对此进行了定义。这一规定需要最高等级稳健性。

注2 – 这些规定确保内容安全系统可要求最弱ECI根状态被应用于在进行任何对安全敏感的操作之前对ECI主机装载的所有项目的验证中。

11.5 客户端图像解密

为了解密一个**ECI客户端**图像，**高级安全系统**可以对由**ECI客户端的运营商**提供的加密的图像解密密钥解密并对[ITU-T J.1012]第7.8条定义的**ECI客户端**图像解密。**ECI客户端**图像解密应在**ECI客户端**图像签名检查之前执行。用于计算图像解密密钥的AS函数为第 8.2.4.12条中定义的reqAsComputeImageKey。**ECI主机**从**运营商**处接收要求的加密密钥信息inputV（发送给计算鉴权密钥的**鉴权机制**的输入信息）、eKey（用鉴权密钥机密的图像解密密钥）和[ITU-T J.1012]第7.8条定义的来自**运营商**的“online”和“min_root_state”参数，并在之后使用提供给客户端的**AS槽**来执行解密（见[ITU-T J.1012]第7.8条）。任何用于图像解密密钥交换会话的当前状态都应该是最新的（值来自重新初始化的**AS槽**）。

12 定时要求

12.1 引言

ECI客户端需要在特定定时限制范围内执行，以满足安全系统（它们是该系统组成部分）的要求。为此，**ECI客户端**依赖于**AS系统**（通过**ECI主机**）提供的函数的某些操作特征。本条定义了**AS系统**函数的计时特征。

AS系统计时特征将函数分成四类：

1) 仅需要AS槽中的管理功能的函数。

2) 只需要对称加密操作的函数，例如使用AK的密钥阶梯计算或解密。

3) 需要在密钥阶梯块或者CPS中的一至四个非对称加密操作的函数，例如LK1的装载和涉及AK计算的执行函数。

4) 需要可能更长的证书链（比如导入/导出链和微客户端鉴权链）的处理的函数。

ECI客户端可通过异步信息引发后三个类别函数。第一个类别的函数可以是同步或者异步的。

非对称加密操作耗费更多时间。任何正在进行的非对称加密操作不得对前两个类别的函数造成拖延。如果一个类别1或2的函数需要类别3或4函数中的操作的结果，**ECI客户端**负责类别3或4中的函数的结果的同步。即，需要等到非对称操作可用（即收到结果信息之后）之后才能引发取决于该结果的函数。

12.2 管理函数

类别1)的函数适用对称和非对称信息的一般标准。

12.3 对称加密函数

调用对称加密操作的函数应由**AS系统**来执行，其基础是每个**AS槽会话**应能在特定时间框架内执行一项功能。一个建议的值可以在[b-ITU-T J Suppl.7]的第6.6.3条中找到。

12.4 非对称加密函数

调用非对称加密操作的函数（例如，涉及密钥阶梯对称密钥计算或使用**鉴权机制**的结果）应由**AS系统**来执行，其基础是每个**AS槽**应能一次执行一项功能。建议的典型值可在[b-ITU-T J Suppl. 7]的第6.6.4条中找到。

附件A

加密函数定义

（本附件是本建议书不可分割的组成部分。）

A.1 散列函数

当前建议书中的散列函数全都基于NIST FIPS PUB 180-4 [NIST FIPS 180-4]中定义的SHA256。

函数散列第7.2条等于[NIST FIPS 180-4]中定义的SHA-256()。

c-函数asHash(uchar *data, uintdatalength, resultLength, uchar *result)使用始于作为*dataIn*八比特字节串的长度dataLength的数据的八比特字节，并计算作为一个resultLength/8八比特字节串的八比特字节串resultOut，并根据以下将其储存在结果中：

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

resultLength应为8的倍数，truncate应为一个比特串（参数1）到长度（参数2）比特的左截断的函数。

BS2OSP和OS2BSP是[ITU-T J.1015]第9条定义的将比特串转换到八比特字节串（反之亦然）的函数。

A.2 非对称加密

非对称加密和解密操作应由[ITU-T J.1015]第10.2和10.3条定义。

A.3 随机数生成

当前建议书中定义的随机数生成应遵守[NIST 800-90Ar1]并满足以下规定：

- 至少在系统启动时（芯片的**AS系统**重启）应生成一个新的安全的独特随机种子数。这一过程取决于芯片或其无法复制且无法被操纵的环境的物理（噪音）或其他属性。生成数的熵应至少为128比特。
- 任何随机数都应使用确定性伪随机数生成器基于上述随机种子数根据[NIST 800-90Ar1]生成。芯片可定期重新设置生成器和/或者如[NIST 800-90Ar1]第8.7条中定义的采用内部（噪音）或难以操纵的外部输入来增加熵。在个性化字符串中应至少使用芯片id。

注 – 在许多AS应用程序中，随机数生成器的实际随机性并非至关重要，随时间推移的唯一性才是关键。这些是典型的现时应用程序：即，用于在解密时重播保护和在内容加密时插入随机数的在线鉴权的随机数。在非对称**微服务器**模式下的加密**AS槽**中作为LK1生成的随机密钥是个例外。

附录I

样本微DRM系统应用

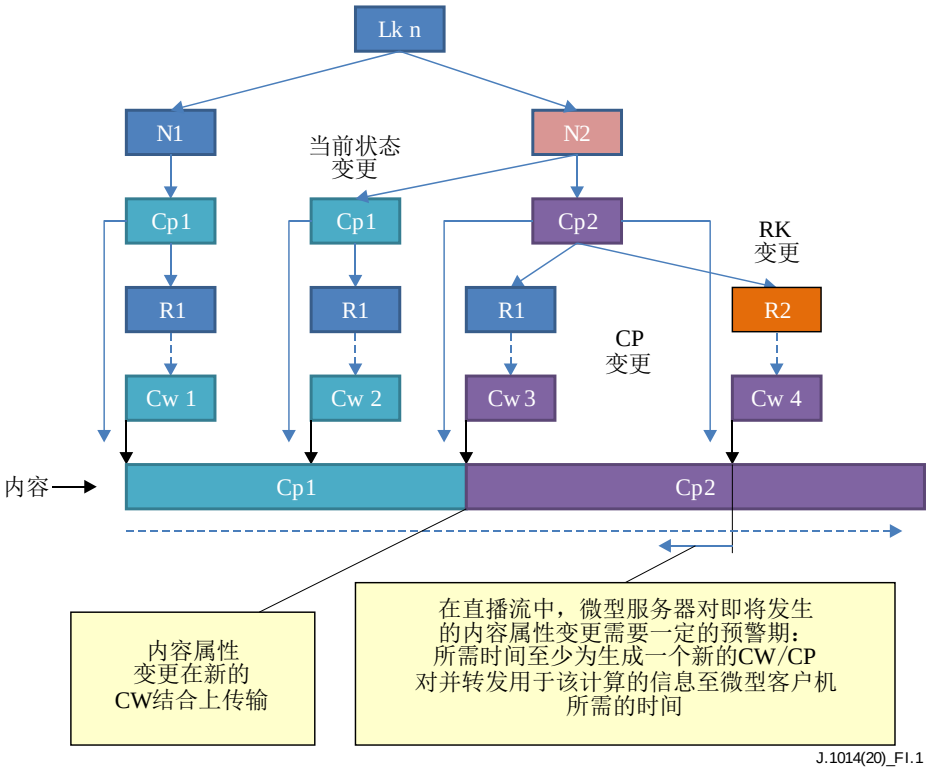
（本附录非本建议书不可分割的组成部分。）

I.1 引言

本附录提供了在一个TS流上操作的**微DRM系统**的执行的**AS系统**的应用的实例。**ECI客户端**的加密以及解密示范操作均得以展示。展示的焦点放在各类行动以及控制字的序列以及需要被生成的相关微DRM信息的并发上（从**微服务器**到**微客户端**，反之亦然）。**微DRM系统**在加密时采用随机密钥生成，在解密时采用现时生成（以防止重放）。假设两个随机密钥都有限值。

I.2 应用情形

图I-1中的应用情形显示了加密侧的**密钥阶梯**状态。LK_n是密钥体系中的第三个却是最低一层密钥。以下是来自**微客户端**的当前状态（N1或N2）、**内容属性**CP1和CP2（在n+2阶段被处理进**密钥阶梯**的输入-C）的和输入**密钥阶梯**n+3阶段的随机密钥种子R1和R2。控制字CW1..CW4从这些**密钥阶梯**输入中得出，并与其相关**内容属性**一起被应用于内容。



图I-1 – 控制字计算密钥体系演进示例

微服务器密钥体系最低的三个阶段的起始状态为N1、CP1和R1。从这些中计算出CW1来为内容加密。触发比特的初始状态为t1。在这个例子中，**微服务器**首先接收一个新的当前状态，并确定是时候将其以CW2的形式应用于未来内容。它首先向**微客户端**发送一个带有新的触发比特t2和加密的密钥组（t2、N2、CP1、R1）的ECM类型的消息，等待一段时间以确定**微客户端**可以接受到并预计算新的控制字CW2，并为即将到来的变更做好准备。之后，它计算新的控制字CW2本身并投入应用，触发相关加密TS流的触发比特中的变更。

图I-1中的下一个事件显示待加密内容的**内容属性**的变更。**微服务器**接收到来自**ECI主机**的信息，告知**内容属性**将变更为CP2。**微服务器**发送带有新加密的密钥组（t3、N2、CP2、R1）的ECM类型信息至其**微客户端**，并从这一新的密钥序列中预计算CW3。当新的**内容属性**被应用时，加密内容上的触发比特自动变更，新的加密控制字CW3机器相关**内容属性**CP2被立即应用。

最后一个事件是**微服务器**决定随机密钥R1变更为R2。该过程与更改一个当前状态的过程基本相同。**微服务器**发送一条带有（t4、N2、CP2、R2）的ECM类型信息至**微客户端**，允许其预计算W4，并利用延迟来确保**微客户端**有足够的时间进行准备。之后使用**阶梯密钥**来计算CW4，并将其应用于内容，导致内容触发比特的状态变成t4。

I.3 猜想和表示法

内容由导出客户端加上相关解密**AS槽**传送至加密**AS槽**的导入连接。导出客户端生成通知在导入内容实际发生之前**内容属性**的任何变化的信息至**ECI主机**。这一过程使用[ITU-T J.1012]中的AS API。

使用以下表示法：

<event-name>(parameters) -><pseudo-code statement> ;表明发生事件时，带有以下伪码的事件名（消息接收）被执行。

以下事件被定义：

- **e_cp(cp):**新的内容属性cp将被用于待加密内容中即将发生的事件（CW变更）。优先于**e_cpe()**。
- **e_cpe():**内容属性变更显著（在有限的时间内到期）。
- **e_cpch():**导入内容的内容属性刚刚变更，在正在使用的控制字不反映这一点时将要求紧急变更。在控制字由于内容属性变更而自动变更时这一事件优先于e_cw()。
- e_nn(nonce):来自**微客户端**的一个新当前状态信息已送达**微服务器**或从**微客户端**发出。
- e_cw():在重新加密内容时触发比特变更，（之前计算的）新CW被应用于内容。
- e_ecm(<parameters>):带有待使用的下个控制字的新参数的信息的接收。
- 事件可以由计时器引发。

cw(toggle_bit,random_key,nonce,content_properties)执行生成用于采用指定参数来加密或解密内容的控制字。在**微服务器**端，首先生成一个拥有相同参数的消息，该消息被转发至**微客户端**，在那里作为e_ecm(…)被接收。

block_cpch() and unblock_cpch()采用信息setAsPermitCPChange(…)来对由于导入内容的**内容属性**变更而导致的加密控制字中的自动变更进行屏蔽或解除屏蔽。

changeCw(toggleBit)在加密侧使用第9.9条定义的信息setAsSC()强制进行控制字（加扰控制字段中的触发比特）转换。

startTimer(timerHandle)启动计时器。

对变量和伪码使用c-类型表示法。

I.4 微服务器伪码

微服务器的密钥混合意味着它需要处理若干同时进行且不同步、可能触发控制字变更的事件：

- 需要新内容属性（由导出客户端的新控制字应用标定至正在被解密的内容）的内容的到来；
- 即将发生的当前状态的失效；和
- 即将发生的随机密钥的失效。

内容属性的变更的处理需要优先级，因为通常在解密过程中的对应控制字变更触发新的内容属性应用之前的时间有限。因此，应足够保守地设置当前状态和随机密钥的失效，因为它们可能因为处理内容属性变更所花费的时间（通常为几秒钟）而推迟。这一情况假设内容属性变更之间的时间始终足够允许至少一个当前状态或随机密钥变更至控制字所需的处理。

当前状态或随机密钥的显著变化的处理有两个优先级。首先，为低优先级设置一个计时器。在**内容属性**没有挂起的变更的情况下，当前状态或随机密钥进行变更，否则为高优先级设置一个计时器。当前状态或随机密钥中的高优先级变更可驳回显著的内容属性变更。但**微服务器**可能得出错误结论。如果发生这种情况，发生在当前状态或随机密钥变更之前的内容属性变更被应用于内容。在这种情况下，必须重新计算一个亦包含新**内容属性**的新控制字。内容属性变更也会在高优先级当前状态或随机密钥变更被应用之后几乎立即发生。在这种情况下反映**微服务器**计算的新的**内容属性**和ECM的CW会推迟。

若TNONCEURGENT和TRKURGENT计时器值可以被设定为一个超过10秒并加上TECM的值，而e_cpch()和e_cw(CPCHANGE)之间的最长时间少于10秒，则可能发生此类冲突，因为任何RK或当前状态变更均可被安排在这样一个周期之后的e_cpch() and和e_cw(CPCHANGE)之间的时间段之前而无需提高优先级。

应注意，以下展示的变量rc和rn的操纵无法由客户端直接进行，而必须使用**AS系统**的函数来执行。

```
/*
   four priority processing model with small shift of CP change time
   in case priority 4 is required (here & now non-anticipated change in CP):
    1) low priority nonce/rk change
    2) low priority CP change (cp eminent but e_cpch() did not occur)
       adopts any previous nonce or rk changes
    3) high priority nonce or rk change; reverts to old CP value
    4) high priority CP change; adopts pending nonce/rk changes and new cp;
       queues new changes

    Optimization may be possible to try to schedule pending nonce and rk changes
    immediately after a CP change; provides modest performance improvement

    State variable invariants/meanings:
    <x> = cp (content property), n (nonce) or r (random key)
    Invariant: p<x> = change in <x> in next CW (p = pending)
                      (not for low priority <n> or <r>)
               q<x> = queud change for <x>, not pending for next CW
               hpcp = high priority content property change (pcp || qcp)
   During a brief time between changeCw() and e_cw() all changes are queued.
   This temporary state is indicated with dhp==true;
 */

#define TECM 3000 /* delay between sending ecm message and changing CW */
#define TNONCEURGENT (2*TEMC + 1000)
#define TRKURGENT (2*TECM + 1000)
#define TNONCE /* some value; may be dynamically determined*/
#define TRK    /* some value; may be dynamically determined*/

toggle(bool t) { return !t }; /* toggles between true and false */

encryptionSession()
/* case rk & nonce change and cp change; unreliable warning cp change (priority with nonce/RK
change) */
```

```
/* first priority on nonce/rk lower than cp, but if urgent it is higher */
{
  SymKey  nc, nn;   /* current and next nonce */
  SymKey  rc, rn;   /* current and next Random Key */
  SymKey  cpc, cpn; /* current and next CP value */
  SymKey  nt, rt;   /* temporary value for nonce, random key */
  TimerHandle t_lpn, t_lpr;
               /* timers for low priority scheduling of nonce and rk change */
  TimerHandle t_n, t_r;
               /* timers for high priority scheduling of nonce and rk change */
  TimerHandle t_ecm_n1, t_ecm_r1;
               /* ecm timers for low priority (1) nonce and rk  ecm */
  Timerhandle t_ecm0, t_ecm1 t_ecm2, t_ecm3;
  TimerHandle t_ecm[4] = {t_ecm0, t_ecm1, t_ecm2, t_ecm3 };
               /* four level 2/3/4 priority level ecm timer pool */
  int t_ecm_cnt = 0;  /* counter for above timer pool allocation */
  bool pn, pr, pcp;   /* true if current CW reflects a change in nonce (nn),
                         random key (rn) or cp (cpn) value */
  bool qn, qr, qcp; /* true if a queued change in nonce, random-key or cp change */
  bool dhp;         /* delay (queue) any new events */
  book hpcp;        /* true if priority 4: high priority CP change */
  int  tCnt1, tCnt234;  /* tCnt<n> is the counter for number of timers
                        in priority <n> that are fired but not yet expired */
  bool t;           /* toggle bit */

/* some macro's are defined to permit reuse of code for processing events */

.* event for next random key */
#define next_r() { rc = rn; rn = rnd128(); startTimer(t_lpr,TRK); }


/* force changeCw on last cascaded higher priority timer unless it is a level 2
   priority cp change in which case the change of CW will be triggered by a CP
   change event */
#define process_emc2_timer(){\
  if (--tCnt234 == 0)\
    if (pn || pr || hpcp){\
      dhp = true; changeCw(toggle(t));\
    } else {\
      /* pcp == true, pn, pr, hpcp == false */\
      unblock_cpch();\
    };\
}

/* on cw-change update state with all processed changes */
#define end_pending() {\
  t = toggle(t);\
  if (pcp) { cpc = cpn; pcp = false };\
  if (pn)  { nc = nn; pn = false };\
  if (pr)  { next_r(); pr = false };\
}

/* move queued events to pending */
#define queued_to_pending() {\
  if (qcp && (!(qn || qr) || cphp)) {
    /* if priority 2 or 4 */\
    pcp = true; qcp = false\
  };\
  /* priority 3 events can be folded with priority 4 */
  if (qn)  { pn = true; qn = false };\
  if (qr)  { pr = true; qr = false };\
}

/* start cw/ecm for pending changes to 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 ;\
  }\
}


  /* only permit auto-changes of toggle bit when prepared */
```

```
block_cpch();

/* receive first cp and nonce values */
for (int i=0; i<2;) {
  ->e_nn(&nc): i++;
  ->e_cp(&cpc): i++;
}

/* initialise state */
pn = pr = pcp = hpcp = false;
dhp = false;
tCnt1 = tCnt2 = 0;
rc = rnd128(); rn = rnd128() ;
t = false;          /* should be initialised to first value in content */
cw(t,rc,nc,cpc) ;   /* will start to be used automatically */

while (!end_session) {
->e_nn(&nn) : startTimer(t_lpn,TNONCE) ;
              /* should occur before nonce limit runs out */
->e_cp(&cpn): /* e.g. compute new export licenses */ ;
->t_lpn()   : { /* low priority nonce change */
    if (pcp || pn || pr || cphp) {
       /* delay new nonce till urgent */
       startTimer(t_n,TNONCEURGENT);
    } else {
      nc = nn;
      cw(t,rc,nc,cpc);
      startTimer(t_ecm_n1,TECM) ;
      tCnt1++;
    }
  };
->t_lpr()   : { /* low priority rk change */
    if (pcp || pn || pr || cphp) {
      /* delay RK till urgent */
      startTimer(t_r,TRKURGENT);
    } else {
      next_r();
      cw(t,rc,nc,cpc);
      startTimer(t_emc_r1,TEMC);
      tCnt1++;
    }
  };
->t_emc_n1() :   /* low priority nonce ecm timer expiry */
->t_emc_r1() : { /* low priority rk ecm timer expiry */
    if (--tCnt1 == 0 && tCnt234 == 0) {
      changeCw();
      dhp = true;
    }
  };
->e_cpe()   : { /* cp change may occur from now on */
    if (dhp || (pn || qn)) qcp = true; /* assert(!hpcp) */
    else { pcp = true; start_pending() };
  };
->t_n()     : { /* urgent nonce change due */
    if (dhp || cphp) qn = true;
    else { pn = true; start_pending() };
  };
->t_r()     : { /* urgent random key change due */
    if (dhp || cphp) qr = true;
    else { pr = true; start_pending() };
  };
->e_cpch()  : { /* high priority change of CP needed */
    cphp = true;
    if (dhp) qcp = true;
    else {
      pcp = true;
      start_pending();
    }
  };
->t_ecm0()  :
->t_ecm1()  :
->t_ecm2()  :
->t_emc3()  : {
    process_timer();
  };
->e_cw()    : { /* assert( pcp && !pn && !pr && !cphp ) */
    end_pending();
    queued_to_pending();
    start_pending();
  };
```

```
  }
}
```

注1 – 在上面提供的**微服务器**例子中，**微服务器**可生成多个触发比特相同的连续但不同的ECM信息，而**微客户端**可接收这些信息。一个极端的例子是第一个e_n()发生，然后在TDELAY e_r()期间发生，接下来在TDELAY之后e_cp(..)发生。在这种情况下，三个连续的ECM信息被发出，触发比特相同，而只有最后一个曾产生被实际应用于内容的控制字。

注2 – 上述代码假设实际触发比特变更总是相对较快地跟随内容属性变更e_cp()之后发生。新的cp值以快得多的速度被获取对**微服务器**没有益处。其生成新的CW的关键触发点是传入内容中的cp–变更事件显著。这触发了用于所有即将进行的CW计算的cp的旧值被新值取代。

在上述示例代码中触发e_n()或e_r()的最短预警时间是e_cp()事件和后续控制字e_cw()的实际变更之间的最坏情况延迟，加上2 × TDELAY加上少量的事件延误和处理时间。

I.5 微客户端伪码

微客户端通过生成两个连续的当前状态信息（用于当前和下一个当前状态）来开启一个会话。若其接收到一个ECM信息，就计算对应控制字。一旦发现发送的最后一个当前状态被应用于一个ECM中，则继续生成并发送一个新的当前状态信息。

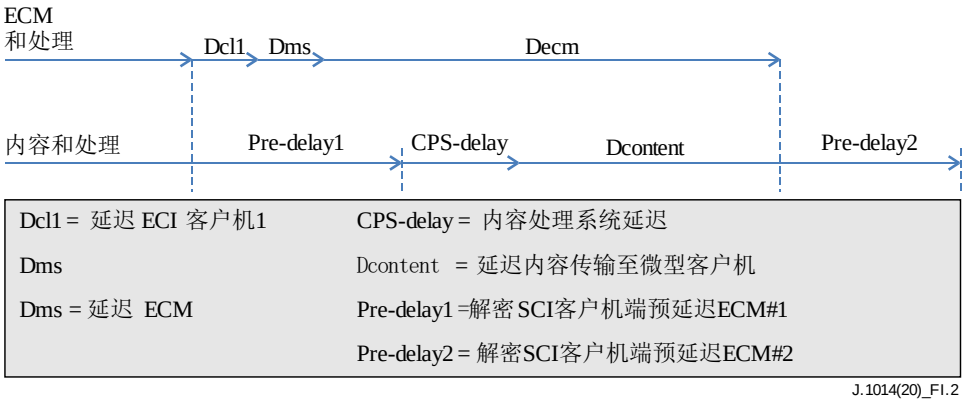
注1 – 安全的当前状态无法由**ECI客户端**代码直接生成，必须使用**AS系统**的适当函数来生成。

```
decryptionSession()
{
  Symkey nc, nn, ln;   /* current, next and last nonce */
  SymKey cp, cpp;  /* received and previous cp */
  SymKey r;        /* received random key */
  bool   t;        /* received toggle bit */
  SymKey n;        /* received nonce */
  bool   end_session; /* end of session reached */


  /* initialise and send nonces */
  nn = rnd128();
  e_nn(nn);
  cpp = Reserved; /* undefined value */
  ln  = Reserved;

  while (!end_session) {
    ->e_ecm(&t,&r&n&cp): {
        if (cp!=cpp) { /* new CP; send event to all export connections via host */
          e_cp(cp);
          cpp = cp;
        }
        cw(t,r,n,cp);
      };
    ->e_cw(): {         /* also triggered on first cw application */
        if (n != ln) {   /* new nonce actually used; move nonce forward */
          nc= nn; nn= rnd128();
          e_nn(nn);
          ln = n;
        }
      };
  } /* end while loop */
} /* end decryption session */
```

注2 – 没有必要从**微服务器**向**微客户端**发送回完整的当前状态值。可以使用一个交替比特作为间接参考。此外，没有严格必要发送所有ECM中的所有参数：只有变更需要被传达至**微客户端**，注意在一些情况下**密钥阶梯**输入当前状态、**内容属性**和随机密钥所有这三个可即刻变更（见第I.4条注2）。发送触发比特对于同步有用，避免任何（有意或无意）重复的ECM信息被作为计算下一个控制字的信息被解译。

I.6 对ECM预延迟的微DRM系统级联效应

微DRM服务器依赖来自发出其导入内容的**微客户端**的即将发生内容属性变更的预警时间（预延迟）来允许它预计算ECM信息并将信息发送至同级**微客户端**。进行所需的处理花费的时间（可能相对较短：通常不需要进行重要计算）加上转发该ECM信息至**微客户端**的时间可能长于用于传递最新重新加密内容至**微客户端**的路径的时间。这意味着**微客户端**经历的新ECM的任何预延迟都比最初导出内容的**ECI客户端**经历的预延迟有了相应缩短。



图I-2 – 预延迟与选择性延迟补偿的时序关系

如图I.2所示，**安全视频路径**可在内容传输中引进一个延迟来补偿转发ECM信息的延迟。这一延迟之后可被选择大致等于延迟偏差。在插入ECM的情况下，Decm和Dcontent紧密匹配。但解密**ECI客户端**中的处理延迟以及**微服务器**中到TS流中的ECM的实际插入点的处理延迟应得到补偿。

I.7 内容属性变更定时接口惯例

正如第I.4条所展示，**微服务器**要求其导入内容中即将发生的内容属性变更的预警。处理这一变更以及发送ECM信息至**微客户端**的最短时间要求的惯例被称为TECM：本附录的以下段落给出了一个例子。在这个例子中TECM的值被设定为3 s。

在级联的**ECI客户端**链中的首个解密**ECI客户端**的最短预警延迟的惯例为TECM + TCASCADE。TCASCADE反映在**微DRM系统**的级联链中的**ECI客户端**处理ECM的最大累积延迟。在这个例子中TCASCADE被设定为2 s。

注1 – 若内容亦被延迟，其用于补偿ECM处理延迟。然而在流模式中这并不可取。

ECI客户端ECM处理延迟的最大值（如第I.6条所述无补偿的Dcl1 + Dms）被称为TDELAY，在本例中被设定为0,3 s。本例子中的值允许6个级联的**微DRM系统**在TCASCADE (2s)范围内操作，使TECM成为**微处理器**最短预警时间。

正如第I.4条所示，为处理传入的内容的内容属性变更而不引发内容属性变更的移位，**微服务器**不仅要求即将发生的内容变更的预警，亦对这一预警期的上限做出要求，从而使其能够安全地处理其他控制字的变更（例如当前状态和随机密钥变更）。在这个例子中，预警期上限TMAXWARN可被安全地设置为10 s。

注2 – 若未遵照惯例，影响可能是一个微DRM系统中的重新加密内容中的内容属性变更的位置的最大TECM移位以及6个微DRM系统级联中的最大TECM+6*TCASCADE移位。

强烈建议足够早地设计当前状态和随机密钥低优先级警告（t_lpn and t_lpr in clause B.4），从而允许**内容属性**中的一个（甚至一些）变更对延迟当前状态和随机密钥变更的处理。如果内容属性变更能够及时地被充分隔开（并且遵守TMAXWARN），可防止在处理当前状态或随机密钥变更中发生超出限度的情况。

定时参数的选择对于ECI客户端之间的内容无缝切换十分重要。[b-ITU-T J.Suppl.7]第6条中提供了关于延迟参数TECM、TCASCADE、TDELAY和TMAXWARN建议值的更多信息。

附录II

有待进一步发展的领域

（本附录非本建议书不可分割的组成部分。）

已经确定，本建议书需要做进一步的开发和验证，才能满足[ITU-T J.1010]中规定的要求，并且[ITU-T J.1010]需要进行更新，以反映MovieLabs增强型内容保护（ECP）规范[b-ECP]的要求。[b-ITU-T J.1011]、[ITU-T J.1012]、ITU-T J.1013、[b-ITU-T J.1014]、[b-ITU-T J.1015]和 [b-ITU-T J.1015.1]建议书未来应予更新，以反映对[ITU-T J.1010]的那些更新。

国际电联的许多成员国以及各行各业的利益攸关方 – 包括设备和电子组件的制造商、受版权保护的内容的所有者和被许可者、机顶盒（OTT）和线性电视服务的提供商以及全球各地基于有条件访问系统（CAS）和数字版权管理（DRM）解决方案的提供商都对嵌入式通用接口（ECI）不能完全满足ECP要求以及更广泛的行业内容保护要求表示了担忧。

更具体地说，其对ITU-T第9研究组（SG9）会议（2020年4月16-23日）的文稿引起了人们的关注。来自以色列、澳大利亚、ITU-T部门成员三星公司以及SG9准成员Sky Group和MovieLabs的文稿提议在ECI建议书中纳入一系列更改，但未就此达成共识。这些项目在[b-SG9 Report 17 Ann.1]中进行了清点。

这些建议包括：

1） 通过缩小ECI范围来简化ECI系统；

2） 取消DRM；

3） 取消对内容的重加密；

4） 取消软件管理；

5） 增加用于安全存储和加密操作的API；

6） 允许供应商特定的密钥阶梯；

7） 采用ITU-T J.1207 TEE要求；

8） 包括VM的TEE实施方案；

9） 升级密码算法的强度，例如，使用SHA-384；

10） 使用标准证书，例如，ITU-T X.509；

11) 重新考虑客户端之间的通信；

12） 与ETSI进行其他联络；

13） 进行额外的同行评审；

14） 探索信托机构模型的替代方案；

15） 进一步定义ECI合规性和稳健性规则的技术方面问题；

16） 增加有关多样性的要求，例如，地址空间随机化；

17） 增加有关运行时完整性检查的要求。

这些建议反映出内容保护及其违背之可能带来的威胁正在不断演变。ECI最初是在批准本ITU-T建议书之前近十年来构思的。对像ECI这样的系统，需要定期根据攻击技术和行业保护要求的最新状况进行评估。

存在其他机制以实现互操作性。特别是对DRM用例，大多数互联网视频服务已经部署了其他解决方案，以提供互操作性并满足其需求。

由于许多成员国将国际电联标准视为对其市场和行业发展有影响力的指导来源，因此进一步的澄清很重要。关注清单确保ECI在其国内市场中的实施，这可涉及对本ITU T建议书含义的全面理解，并确保在考虑立法、法规或有关要求消费者数字电视设备可互操作的市场需求时能虑及这些问题。它还确保技术设备制造商（它们可能更喜欢使用一组独特的要求或其他标准来设计产品）在为不同市场开发产品时可以考虑到这些问题。

参考资料

[b-ITU-T J.1015.1] ITU-T J.1015.1建议书（2020年），用于可交换有条件访问/数字版权管理（CA/DRM）解决方案的嵌入式通用接口（ECI）；高级安全系统 – 密钥阶梯块：控制字使用规则信息和相关数据1的鉴权。

[b-ITU-T J-Suppl.7] ITU-T 系列J建议书 – 增补7（2020年），用于可交换有条件访问/数字版权管理（CA/DRM）解决方案的嵌入式通用接口（ECI）；ECI实施导则（EG）。

[b-SG9 Report 17 Ann.1] ITU-T SG9会议报告，SG9-R17-附件1（2020年），2020年4月16-23日召开的SG9全虚拟会议第17号报告附件1。
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 Available at:
https://movielabs.com/ngvideo/MovieLabs_ECP_Spec_v1.2.pdf

ITU-T 系列建议书

系列 A	ITU-T 工作的组织
系列 D	资费及结算原则和国际电信/ICT 的经济和政策问题
系列 E	综合网络运行、电话业务、业务运行和人为因素
系列 F	非话电信业务
系列 G	传输系统和媒介、数字系统和网络
系列 H	视听及多媒体系统
系列 I	综合业务数字网
系列 J	**有线网络和电视、声音节目及其他多媒体信号的传输**
系列 K	干扰的防护
系列 L	环境与 ICT、气候变化、电子废物、节能；线缆和外部设备的其他组件的建设、安装和保护
系列 M	电信管理，包括 TMN 和网络维护
系列 N	维护：国际声音节目和电视传输电路
系列 O	测量设备的技术规范
系列 P	电话传输质量、电话设施及本地线路网络
系列 Q	交换和信令，以及相关的测量和测试
系列 R	电报传输
系列 S	电报业务终端设备
系列 T	远程信息处理业务的终端设备
系列 U	电报交换
系列 V	电话网上的数据通信
系列 X	数据网、开放系统通信和安全性
系列 Y	全球信息基础设施、互联网协议问题、下一代网络、物联网和智慧城市
系列 Z	用于电信系统的语言和一般软件问题

瑞士印刷
2020年，日内瓦