GY

中华人民共和国广播电影电视行业标准

GY/T 255—2012

可下载条件接收系统技术规范

Technical specification of downloadable conditional access system

2012－03－16 发布　　2012－03－16 实施

国家广播电影电视总局　　发布

目　　次

前言 III
引言 IV
1　范围 1
2　规范性引用文件 1
3　术语、定义、缩略语和约定 1
3.1　术语和定义 1
3.2　缩略语 3
4 概述 3
5 安全机制 4
5.1　密钥机制 5
5.2　DCAS 终端软件安全机制 6
5.3　DCAS 终端硬件安全机制 6
6 总体要求 7
6.1　DCAS 前端系统要求 7
6.2　安全芯片密钥植入模块要求 7
6.3　DCAS 用户端软件要求 7
6.4　终端安全芯片要求 7
6.5　终端软件平台要求 8
6.6　终端安全和鲁棒性要求 8
7 系统功能概述 8
7.1　DCAS 系统结构 8
7.2　DCAS 前端模块功能概述 9
7.3　终端模块功能概述 11
7.4　DCAS 安全数据管理平台功能概述 11
8 终端系统 11
8.1　终端系统架构 11
8.2　DCAS 应用程序接口 12
8.3　DCAS 用户端软件下载及启动 12
8.4　典型的终端发放流程 13
9 终端安全芯片 13
9.1　安全芯片工作流程 13
9.2　根密钥派生模块 15
9.3　层级密钥 15
9.4　OTP 区域 17
附录 A（规范性附录）终端启动加载软件启动机制 18
附录 B（资料性附录）DCAS 用户端软件签名、验签过程 21
附录 C（规范性附录）DCAS 应用程序接口 31

附录 D（资料性附录）DCAS 用户端软件下载和启动的机制 53
附录 E（资料性附录）DCAS 终端生产发放使用和软件签名认证流程 56
附录 F（规范性附录）终端安全芯片功能扩展..... 59
附录 G（资料性附录）根密钥派生模块相关要求..... 62
附录 H（规范性附录）握手认证流程..... 63

前　言

本标准按照GB/T 1.1-2009给出的规则起草。

本标准由全国广播电影电视标准化技术委员会（SAC/TC 239）归口。

本标准起草单位：中央电视台、国家广播电影电视总局广播科学研究院、北京永新视博数字电视技术有限公司、北京数码视讯科技股份有限公司、北京安视网信息技术有限公司、东方有线网络有限公司、华数数字电视传媒集团、江苏省广播电视信息网络股份有限公司、国家广播电视网工程技术研究中心、国家网络新媒体工程技术研究中心、北京数字太和科技有限责任公司、天栢宽带网络科技（上海）有限公司、武汉天喻信息产业股份有限公司、美博通通信技术（上海）有限公司、耐格如信（上海）软件技术服务有限公司、爱迪德技术（北京）有限公司、深圳市同洲电子股份有限公司、山东泰信电子有限公司、意法半导体研发(上海)有限公司、湖南神峰电子科技有限公司、泰鼎多媒体技术（上海）有限公司。

本标准主要起草人：丁文华、盛志凡、吴敏、李伟东、宿玉文、李望舒、万乾荣、邹峰、杨勍、解伟、王明敏、张卫、孙庭、张晶、熊彬、田雪冰、王强、储原林、孟祥昆、饶丰、郑力铮、尹又兴、王茵、梁建、王劲林、王兴军、梅红兵、彭冲、余斌、毛展来、殷毅波、李晖、张选东、常毅、宋文强、刘正华、程国海、周俊仁、谢兆安、于立峰。

引　言

本标准的发布机构提请注意如下事实，使用者声明符合本标准时，可能使用涉及本标准有关内容的相关授权的和正在申请的专利。

本标准的发布机构对于专利的真实性、有效性和范围无任何立场。

专利持有人已向本标准的发布机构保证，愿意同任何申请人在合理和无歧视的条款和条件下，就专利授权许可进行谈判。该专利持有人的声明已在本标准的发布机构备案。

相关信息可以通过以下联系方式获得：

专利权利人	联系地址	联系人	邮政编码	电话	电子邮件
中央电视台	北京市复兴路11号	蔡贺	100859	010-68506440	caihe@cctv.com
国家广播电影电视总局广播科学研究院	北京市复兴门外大街2号	杨勍	100866	010-86094200	yangqing@abs.ac.cn
北京永新视博数字电视技术有限公司	北京市海淀区上地东路5号京蒙高科大厦B座4层	田雪冰	100085	010-62971199	tianxb@novel-supertv.com
北京数码视讯科技股份有限公司	北京市海淀区上地五街数码视讯大厦	熊彬	100085	010-82345800	xiongbin@sumavision.com
北京安视网信息技术有限公司	北京市朝阳区光华路4号东方梅地亚中心A座2501-2506	梁建	100026	010-85279588	lliang@nds.com

请注意除上述专利外，本标准的某些内容仍可能涉及专利。本标准的发布机构不承担识别这些专利的责任。

可下载条件接收系统技术规范

1 范围

本标准规定了可下载条件接收系统的总体要求、安全机制、系统架构和功能、终端系统、终端安全芯片等内容。

本标准适用于具有双向交互能力的广播电视网的可下载条件接收系统。

2 规范性引用文件

下列文件对于本标准的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本标准。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本标准。

GY/Z 175-2001　数字电视广播电视条件接收系统规范

无线局域网产品使用的SMS4密码算法　国家密码管理局，2006年1月

SM2椭圆曲线公钥密码算法　国家密码管理局，2010年12月

SM3密码杂凑算法　国家密码管理局，2010年12月

RFC 3447　公钥密码标准#1：RSA密码规范2.1版（Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1）

FIPS PUB 180-4　安全散列标准（Secure Hash Standard (SHS)）

FIPS PUB 197　高级加密标准（Advanced Encryption Standard (AES)）

NIST SP 800-67 Revision 1　三重数据加密算法分组加密建议（Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher）

3 术语、定义、缩略语和约定

3.1 术语和定义

下列术语和定义适用于本标准。

3.1.1

安全芯片密钥还原函数　secure chipset key de-obfuscation

用于将加密的安全芯片密钥还原成安全芯片密钥的算法。

3.1.2

层级密钥　key ladder

一种保证控制字安全传送的结构化多级密钥机制。

3.1.3

传输流过滤　transport stream filtering

通过一定的过滤机制，从传输流中提取符合过滤条件的数据。

3.1.4

复用　multiplex

在一个物理频道中承载一个或多个业务或事件的所有数据流。

3.1.5

控制字　control word

用于加解扰的控制信息。

3.1.6

加扰　scramble

加扰是为了保证传输安全而对业务码流进行特殊处理。通常在广播前端的条件接收系统控制下改变或控制被传送业务码流的某些特性，使得未经授权的接收者不能得到正确的业务码流。

3.1.7

解扰　descramble

解扰是加扰的逆过程。

3.1.8

根密钥　root key

用于层级密钥首层的密钥。

3.1.9

规避手段　circumvention method

用于规避本标准所要求的安全机制中任一功能的方法，包括但不限于硬件、软件或软硬件组合。

3.1.10

哈希值　hash value

使用Hash算法对任一数值进行计算后得到的值为哈希值。

3.1.11

派生　derivation

使用相同输入数据应用不同参数，生成不同输出数据的单向演化过程。

3.1.12

启动加载软件　bootloader

用于接收终端启动后初始化硬件和加载主软件的程序

3.1.13

随机数　nonce

由DCAS前端发出的用于握手认证的随机或非重复值。

3.1.14

握手认证　challenge response

DCAS前端验证终端安全芯片合法性的一种流程。

3.1.15

终端安全芯片　terminal secure chipset

集成在接收终端中，主要实现密钥派生和层级密钥的码流处理芯片。

3.1.16

终端安全功能模块　terminal secure module

集成在接收终端中,通过终端安全芯片和可选的智能卡等可分离安全设备配合终端软件完成层级密钥处理、解密、解扰等条件接收功能的安全硬件模块。

3.1.17

终端软件平台　terminal software platform

运行于接收终端上，底层集成了各项硬件访问和驱动，向上层包括CA应用在内的各项终端应用提供API，并对上层应用提供按一定安全要求进行应用下载、启动和运行环境的软件平台。

3.1.18

最终根密钥派生函数　final root key derivation function

用于生成根密钥的一种派生算法。

3.2　缩略语

CA	条件接收（Conditional Access）
CAT	条件接收表（Conditional Access Table）
CAS	条件接收系统（Conditional Access System）
ChipID	芯片标识（Chipset Identification）
CPU	中央处理器（Central Process Unit）
CW	控制字（Control Word）
DCAS	可下载条件接收系统（Downloadable Conditional Access System）
ECM	授权控制信息（Entitlement Control Message）
ECW	加密的控制字（Encrypted Control Word）
EMM	授权管理信息（Entitlement Management Message）
EPG	电子节目指南（Electronic Program Guide）
ESCK	加密的安全芯片密钥（Encrypted Secure Chipset Key）
OTP	一次性可编程（One Time Programmable）
PID	包标识（Packet Identification）
SCK	安全芯片密钥（Secure Chipset Key）
SCKv	安全芯片密钥厂商派生密钥（Secure Chipset Key Vendor）
Seedv	掩码密钥厂商派生密钥（Seed Vendor）
SI	业务信息（Service Information）
SMK	掩码密钥（Secret Mask Key）
Vendor_SysID	条件接收供应商系统标识（Vendor System Identification）

4　概述

本标准规范了DCAS系统架构、功能模块和接口等技术要求，定义了DCAS的密钥和安全机制，明确

了 DCAS 用户端软件下载的安全信任链机制和安全下载流程，描述了 DCAS 终端生产发放使用和软件签名认证流程等。

DCAS 由前端、终端和安全数据管理平台组成，包括 7 个功能模块：1）DCAS 前端；2）DCAS 用户端软件；3）终端安全芯片；4）终端软件平台的 DCAS 应用程序接口；5）安全数据管理平台； 6）安全芯片密钥植入模块；7）可分离安全设备（本标准中暂不做定义）。DCAS 架构图见图 1。

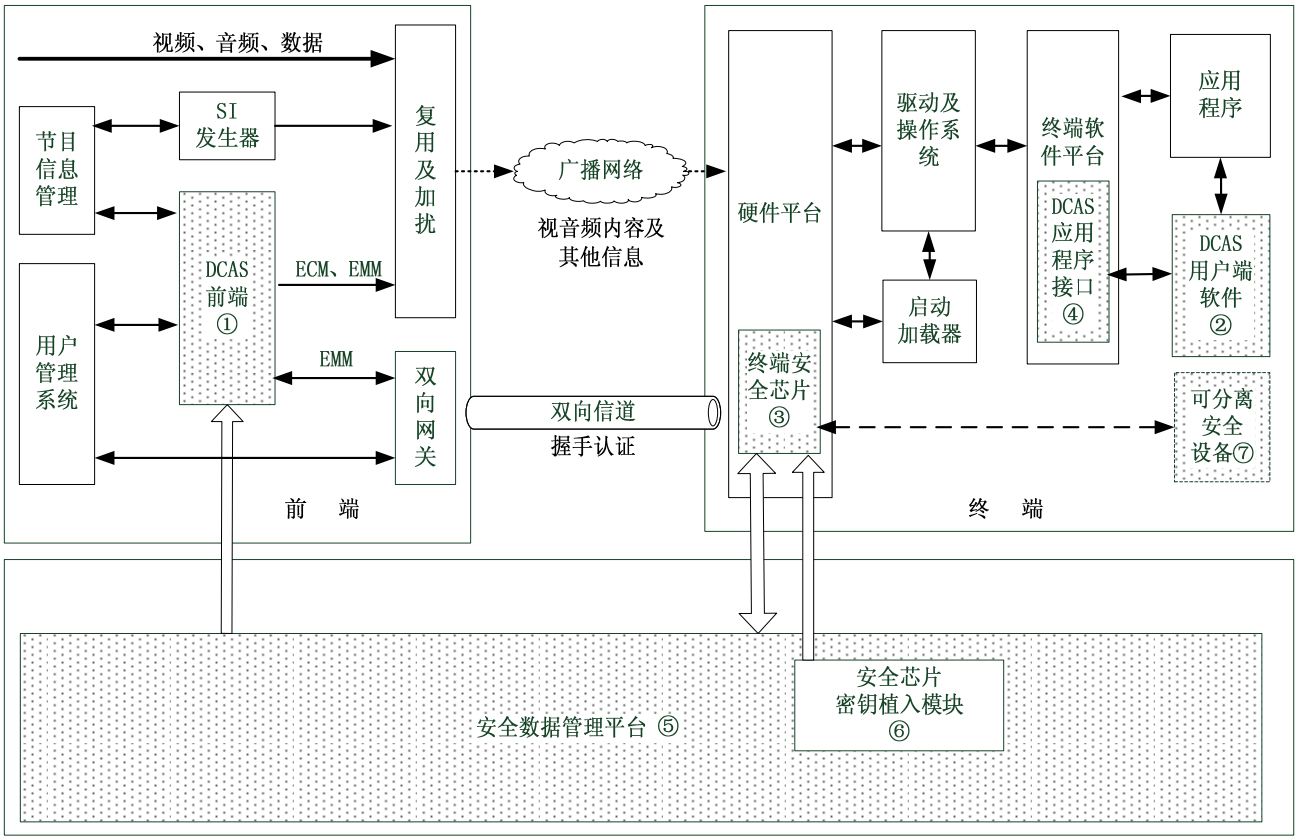


图1 DCAS 架构图

DCAS 是一套完整的端到端业务保护系统，具有传统条件接收系统所有的授权控制和管理功能，能够同时支持 DCAS 终端和传统条件接收终端，并可以通过双向信道和广播信道对终端进行授权，授权效率高、系统安全性好。接收终端可以通过 DCAS 用户端软件下载，实现在不同 DCAS 系统终端间的灵活切换，从而实现终端业务保护水平化。

DCAS 密钥机制由根密钥派生、层级密钥、安全数据管理等机制构成。DCAS 根密钥派生机制使不同 DCAS 系统能够基于同一终端安全芯片派生出个性化根密钥，从而使 DCAS 终端能够根据所下载的 DCAS 用户端软件实现对相应 DCAS 前端加密内容的解密；DCAS 层级密钥机制通过握手认证功能实现 DCAS 终端安全芯片、DCAS 用户端软件和 DCAS 前端的相互鉴真和互信，保障了 DCAS 层级密钥的密钥安全传输和处理；DCAS 安全数据管理机制运用数据分散安全管理的方法，保障了 DCAS 系统的安全性和中立性。

DCAS 安全机制综合运用 DCAS 密钥机制、软硬件安全处理技术和安全管理手段，对 DCAS 前端、终端安全芯片和用户端软件以及安全芯片密钥植入模块进行安全保护，保证端到端系统安全。

5 安全机制

5.1 密钥机制

5.1.1 密钥模型

DCAS 系统的密钥机制包括根密钥派生机制、层级密钥机制、安全数据管理机制和业务加解扰机制。DCAS 系统的密钥模型见图 2。

根密钥派生机制使安全芯片能实时运用其内置的根密钥派生模块派生出个性化根密钥；安全数据管理机制通过数据分散安全管理保障了根密钥生成管理的安全性；根密钥派生机制和安全数据管理机制密切协同，使不同 DCAS 系统可安全地基于同一安全芯片的根密钥派生模块派生出各自个性化的根密钥。

层级密钥机制实现控制字在传输过程中的保护，是以根密钥派生机制派生获得的根密钥为第一级密钥，对控制字进行逐层加解密保护，同时层级密钥中还包含握手认证功能以实现对终端安全芯片合法性的认证。对内容的保护是通过采用层级密钥机制输出控制字实现的业务加解扰机制。

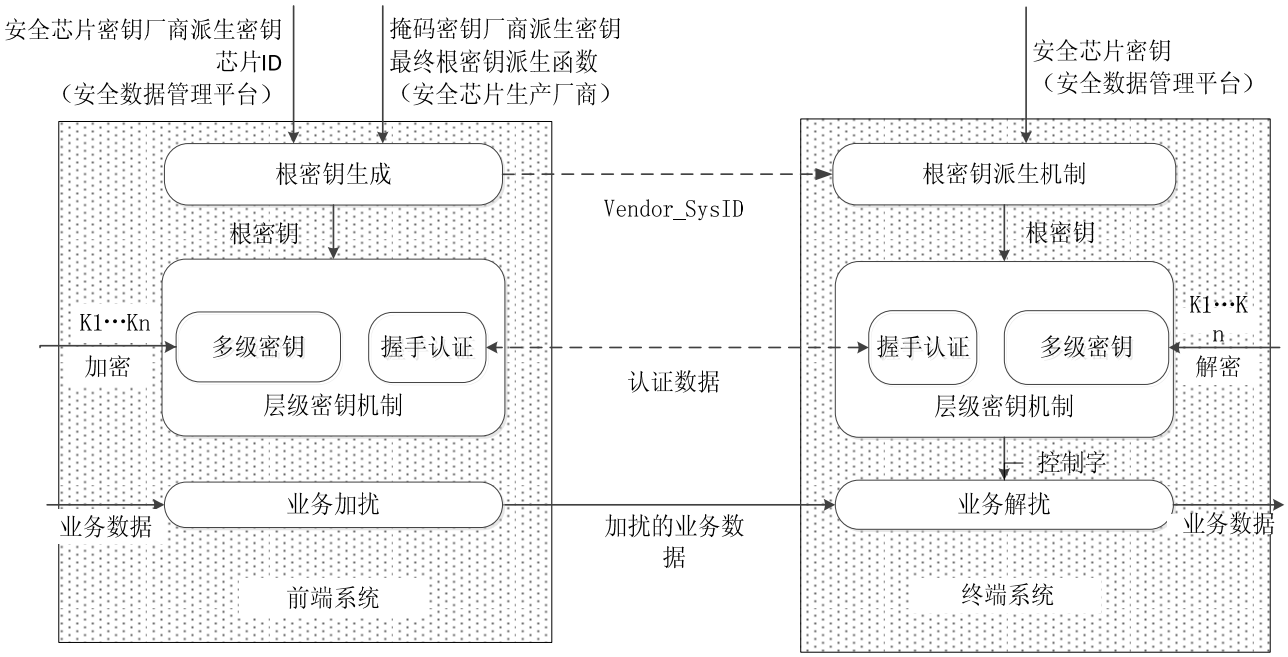


图2 密钥模型

5.1.2 根密钥派生机制

根密钥派生机制包括前端根密钥派生和终端根密钥派生，根密钥派生的流程见 9.2。

a) 前端根密钥派生

DCAS 前端系统采用终端安全芯片生产厂商和安全数据管理平台提供的必要数据和相关算法，在终端使用前为每个终端派生出对应安全芯片的根密钥。

b) 终端根密钥派生

DCAS 终端利用 DCAS 用户端软件，为安全芯片提供相关必要信息，并通过安全芯片内预置的根密钥派生模块派生出根密钥。

5.1.3 层级密钥机制

层级密钥机制包含多层密钥功能和握手认证功能。

多层密钥功能是指按照层级方式对控制字进行逐层加密，保证控制字在使用和传输过程的安全。

DCAS 前端系统生成密钥对控制字逐层加密，终端安全芯片接收密钥逐层解密还原控制字，终端安全芯片的多层密钥具体技术要求见 9.3.1。

握手认证功能即是在双向系统中对终端安全芯片进行的合法性验证。终端安全芯片对接收到的认证数据采用层级密钥中的密钥进行处理并返回结果，DCAS 前端对终端安全芯片返回的结果进行认证。具体技术要求见 9.3.2。

5.1.4 安全数据管理机制

安全数据管理机制是 DCAS 密钥机制的核心组成部分。它运用安全信息分散管理的方法，将前端和终端根密钥派生中所需的信息交由安全数据管理平台、条件接收厂商和芯片厂商分别管理，保证了 DCAS 系统的安全性及中立性。

5.1.5 业务加解扰机制

业务加解扰机制实现业务数据从前端到终端的安全传递。广播节目加解扰机制具体技术要求见 GY/Z 175-2001。

5.2 DCAS 终端软件安全机制

5.2.1 信任链

DCAS 用户端软件安全机制基于一种自底向上的信任链，从终端安全芯片至启动加载软件、终端软件平台和 DCAS 用户端软件，采用数字签名技术建立信任链，只有在在信任链中的每个环节都通过签名校验后，信任链的后一环节方可启动。DCAS 用户端软件除应通过签名安全验证外，在运行时还应有数据安全保证机制。

5.2.2 启动加载软件验证

启动加载软件在运行之前由安全芯片对其进行数据来源可靠性验证和数据完整性验证。

安全数据管理平台负责管理启动加载软件的签名。具体技术实现见附录 A。

5.2.3 终端软件平台验证

终端软件平台下载和运行之前由启动加载软件在本地对其进行数据来源可靠性验证和数据完整性验证。

5.2.4 DCAS 用户端软件验证

DCAS 用户端软件在下载和运行之前需要由终端软件平台对其进行数据来源可靠性验证和数据完整性验证。具体技术实现参见附录 B。

下载的 DCAS 用户端软件需按照运营商下发的应用访问权限文件访问终端资源。具体技术要求见 8.3。

5.2.5 DCAS 用户端软件数据安全

DCAS 用户端软件在终端存储数据时，终端要保证数据存储的安全，避免因为异常断电等因素引起数据丢失。

重要数据加密存储时，可以使用握手认证机制的加密部分对数据进行加密。

5.3 DCAS 终端硬件安全机制

DCAS 终端硬件安全机制通过终端安全芯片保证。具体技术要求见第 9 章。

6 总体要求

6.1 DCAS 前端系统要求

DCAS 前端应符合 GY/Z 175-2001 中第 6 章的要求，同时应支持回传数据的接收和处理，其要求为：

a) 应支持通过双向信道实现对终端的合法性检查；

b) 应支持通过双向和单向信道实现 EMM 传输等功能；

c) 应能生成终端根密钥；

d) DCAS 前端生成的 EMM 和 ECM 应符合 6.2。

6.2 安全芯片密钥植入模块要求

安全芯片密钥植入模块负责将派生根密钥所需的 ChipID、ESCK、BL_KEY0 等必要信息植入终端安全芯片中，在安全芯片生产线上使用，其要求为：

a) 应具有足够的安全性和防攻击能力；

b) 应支持安全存储敏感数据；

c) 应支持 ChipID、ESCK、BL_KEY0 等必要信息的导入和导出。

6.3 DCAS 用户端软件要求

DCAS 用户端软件是 DCAS 中可下载、可替换的终端软件模块，其要求为：

a) 应具有足够的安全性和防攻击能力；

b) 应实现和终端软件平台之间的标准应用程序接口；

c) DCAS 用户端软件通过和终端软件平台交互实现条件接收功能。

DCAS 用户端软件应符合 GY/Z 175-2001 中 7.2.1 的要求,同时应支持以下功能：

a) DCAS 用户端软件下载

终端启动运行后，如果没有在本机的 Flash 内发现已有的 DCAS 用户端软件，终端应通过特定的指令或图形界面下载 DCAS 用户端软件。下载的形式应包含以下两种：用户自愿下载、运营商强制下载。

b) DCAS 用户端软件注册

DCAS 用户端软件应具有向 DCAS 前端系统注册的能力，报告 DCAS 用户端软件的运行状态并获取最新的相关信息。

c) DCAS 握手认证

DCAS 用户端软件应支持终端第一次安装和更换 DCAS 用户端软件时必须使用握手认证。

6.4 终端安全芯片要求

终端安全芯片是集成在接收终端中，实现层级密钥的码流处理芯片，其要求为：

a) 应支持通过派生方式生成根密钥。

b) 应支持层级密钥方式，保证控制字在芯片内的安全传输。

c) 应包括符合 MPEG-2、MPEG-4、H.264 等标准的码流处理模块。

d) 应包括符合 DVB 标准的解扰模块。

e) 不论终端安全芯片是否包含主 CPU，主 CPU 都不得读写存储相关安全信息及密钥的寄存器。

f) 应包含 OTP 区域，并提供根密钥生成及安全数据管理平台根证书的存储功能。

g) 应支持代码签名验证功能。

h) 应支持以下几种标准的加密算法：

   1) AES 算法，遵循 FIPS PUB 197；

   2) Triple-DES (TDES)算法，遵循 NIST SP 800-67 Revision 1；

3) RSA 算法，遵循 RFC 3447；

4) Hash 算法 SHA-1、SHA-256，遵循 FIPS PUB 180-4。

i) 可支持但不限于以下几种标准的加密算法：

1) SMS4 算法，遵循无线局域网产品使用的 SMS4 密码算法；

2) SM2 算法，遵循 SM2 椭圆曲线公钥密码算法；

3) Hash 算法 SM3，遵循 SM3 密码杂凑算法。

6.5 终端软件平台要求

终端软件平台是在终端安全芯片硬件及驱动之上的公共软件模块，其要求为：

a) 应支持 DCAS 用户端软件的下载、更新和替换。

b) 应提供 DCAS 用户端软件所需的标准应用程序接口。

c) 应保证 DCAS 用户端软件下载、启动、运行过程中的完整性、可靠性和安全性。

d) 应可同时支持运行两个以上 DCAS 用户端软件。

6.6 终端安全和鲁棒性要求

6.6.1 安全要求

a) 终端的保密性和完整性

终端设计及生产应能有效地保护本标准中涉及的密钥、证书和软件的保密性和完整性。

b) 终端的安全性

终端生产应有效地保证终端不得被修改或破坏。

终端使用过程中不能通过硬件或者软件的方式停止或者关闭本标准所规定的业务保护功能，从而使终端中的解密内容可以被输出、拦截、转发或复制。

6.6.2 鲁棒性要求

终端的生产或开发应遵循以下要求，从而有效地保护业务安全。

a) 软件

终端在使用软件进行业务保护功能时，该软件应采取安全技术手段保证自身安全，安全技术手段包括软件自身加密、监控执行、在安全物理硬件中实现和执行等。

软件执行过程中，应使用混淆技术，避免软件实现方式被窃取或泄露。

软件应具有完整性检查功能，任何未经授权的修改将导致软件无法运行。

b) 硬件

终端应采用安全技术手段保证硬件的完整性。

终端安全芯片不能通过普通工具或很难通过专业工具被移除、更换；终端安全芯片内数据不得通过任一手段修改。

c) 软硬件接口

终端硬件和软件接口应保证无法通过使用通用工具或很难通过使用专业工具进行破坏或规避。

7 系统功能概述

7.1 DCAS 系统结构

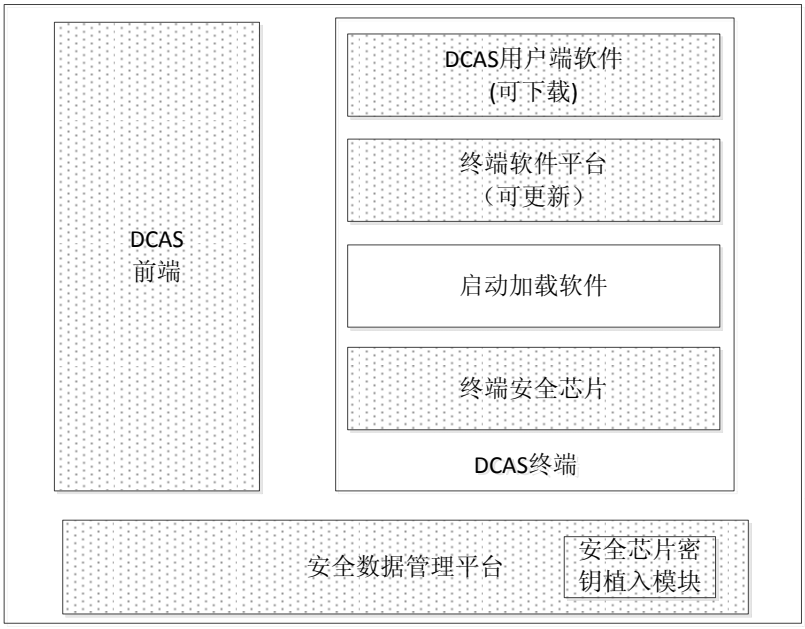


图3　DCAS 系统功能模块图

DCAS 系统功能模块图见图 3，包括 DCAS 前端、DCAS 终端和安全数据管理平台三个部分：

a）DCAS 前端

DCAS 前端对输入的音视频流进行加扰，通过广播信道和双向信道发送条件接收的授权等信息，完成业务的加密保护传送和合法授权控制管理，是实现各项业务和服务的基础。DCAS 前端主要包括 ECMG、EMMG、密钥管理、下载管理和其他模块等。

b）DCAS 终端

DCAS 终端对用户的授权进行合法性验证，解扰受保护的业务，实现业务的条件接收。终端软件平台可以安全地下载、更新和替换 DCAS 终端软件。

c）安全数据管理平台

DCAS 安全数据管理平台负责向终端安全芯片生产厂商提供 ChipID、ESCK、BL_KEY0 等必要信息，向 DCAS 前端提供 SCKv 和 Vendor_SysID 等必要信息。

安全芯片密钥植入模块负责将派生根密钥所需的 ChipID、ESCK、BL_KEY0 等必要信息植入终端安全芯片中，在安全芯片生产线上使用。

7.2　DCAS 前端模块功能概述

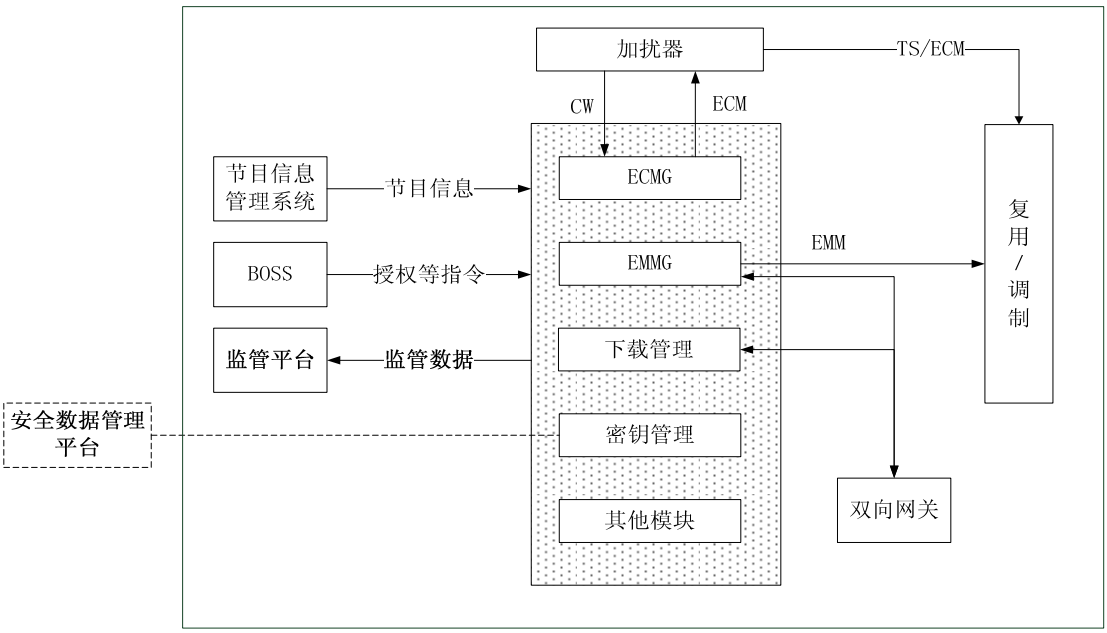


图4　DCAS 前端功能结构图

DCAS 前端功能结构图见图 4，主要模块包括：

a)　ECMG

ECMG 实现和加扰器连接，ECMG 接收加扰器送来的 CW，生成 ECM 信息返回给加扰器，基本功能应满足 GY/Z 175-2001 中附录 E 同密技术中的要求。

b)　EMMG

EMMG 生成 EMM 信息，并通过与复用器接口或双向网关接口发送 EMM。EMMG 的基本功能应满足 GY/Z 175-2001 中附录 E 同密技术中的要求。

c)　密钥管理

负责根密钥和各级密钥的管理。根密钥的生成需要由安全数据管理平台提供的 ChipID、SCKv 和由终端安全芯片厂商提供的 Seedv 和最终根密钥派生函数模块，层级密钥的生成和管理由 DCAS 前端自行处理。

d)　其他模块

DCAS 其他模块可包括：网管模块、监管模块、对外接口等。

e)　下载管理

下载管理结构图见图 5。

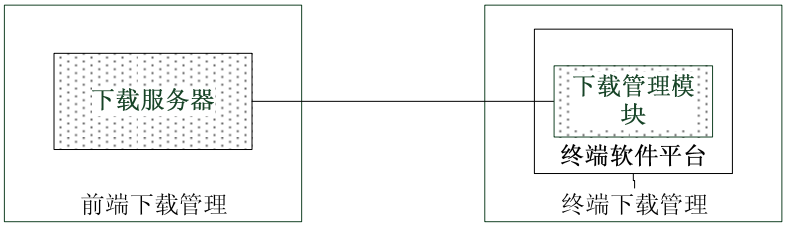


图5　下载管理结构图

前端下载管理模块负责将 DCAS 终端用户软件存储在下载服务器上。

终端下载管理模块是终端软件平台的一个功能模块，负责对 DCAS 终端用户软件进行验证。

下载通道的协议本标准不做定义。

7.3 终端模块功能概述

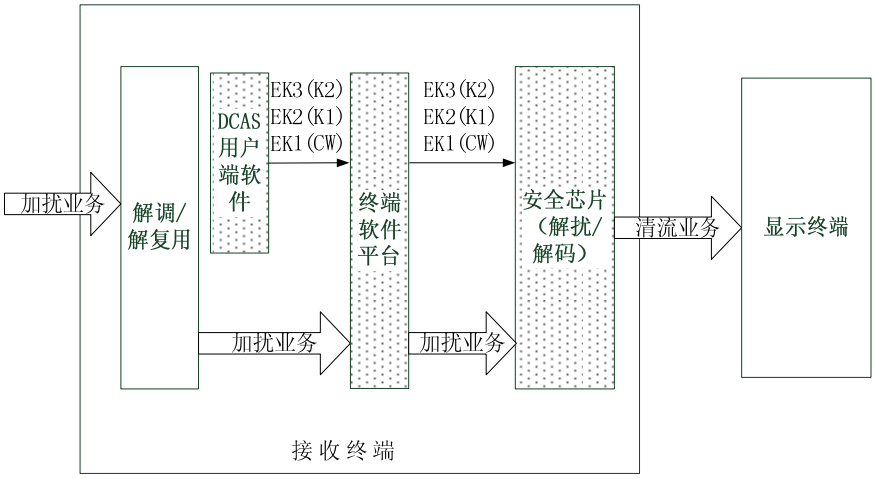


图6 DCAS 终端功能结构图

a) DCAS 终端功能结构见图 6，主要模块包括：终端安全芯片

终端安全芯片接收终端软件平台输入的层级密钥数据和加扰业务，对加扰业务进行解扰和解码，并输出解码后的视音频。

b) 终端软件平台

终端软件平台是在终端安全芯片硬件及驱动之上的公共软件模块。提供 DCAS 用户端软件所需的标准应用程序接口，保证 DCAS 用户端软件下载、启动、运行过程中的完整性、可靠性和安全性。

c) DCAS 用户端软件

DCAS 用户端软件是 DCAS 中可下载、可替换的终端 CAS 软件模块，应具有足够的安全性和防攻击能力，实现和终端软件平台之间的标准应用程序接口。

7.4 DCAS 安全数据管理平台功能概述

a) DCAS 安全数据管理平台

生成 ESCK 功能：生成 ChipID 和 ESCK 文件，通过安全的方式送到安全芯片密钥植入模块，用于把密钥写入安全芯片中。

生成 SCKv 功能：生成 ChipID 和 SCKv 文件，通过安全的方式送到条件接收厂商，用于生成根密钥。

管理各种参数和所有密钥。

b) 安全芯片密钥植入模块

保存安全数据管理平台发来的参数和密钥，放置在芯片厂商生产线上，根据各芯片生产线上的系统通讯，向安全芯片中写入参数和密钥。

8 终端系统

8.1 终端系统架构

DCAS 终端系统架构见图 7，由硬件平台（包括终端安全芯片）、驱动及操作系统、终端软件平台、DCAS 用户端软件及其它应用程序等终端组件组成。本标准仅规定 DCAS 涉及的终端组件：终端安全芯片、DCAS 应用程序接口和 DCAS 用户端软件。

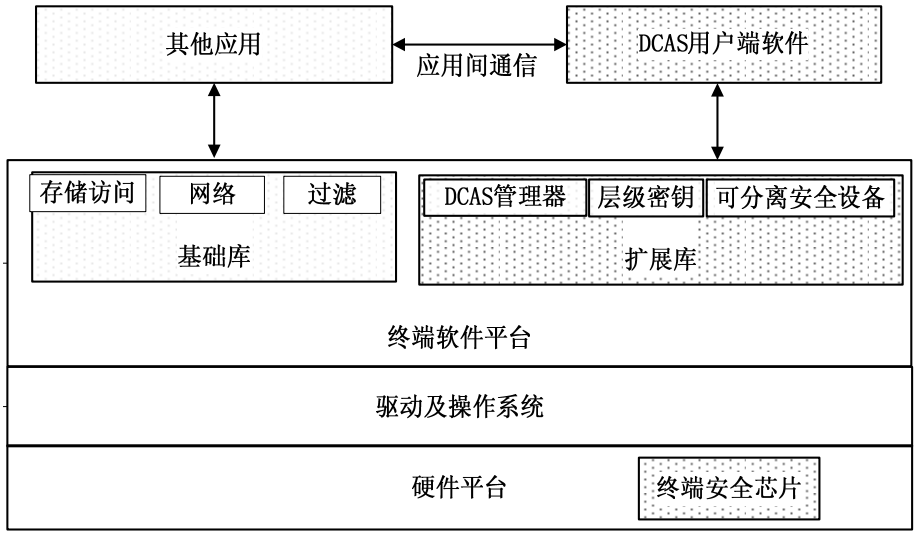


图7　DCAS 终端系统架构

a)　终端安全芯片

终端安全芯片提供层级密钥模块和根密钥生成模块保证终端传输数据安全性和条件接收系统的独立性，具体实现参见第 9 章。

b)　DCAS 应用程序接口

DCAS 应用程序接口实现对 DCAS 用户端软件的管理、支撑 DCAS 用户端软件实现 ECM/EMM 等 CA 数据处理以及实现 DCAS 用户端软件与 EPG 等应用之间的数据交互。

DCAS 应用程序接口支持基础库和 DCAS 扩展库，基础库包含存储访问、网络及过滤模块，DCAS 扩展库包含层级密钥、可分离安全设备、应用间通信及 DCAS 管理器模块。

c)　DCAS 用户端软件

DCAS 用户端软件实现对 ECM、EMM 等数据解析和处理，其接口描述见附录 C。

DCAS 用户端软件通过打包成一个应用后，下载到终端软件平台上，与其它终端软件平台应用并存。

8.2　DCAS 应用程序接口

DCAS 应用程序接口实现 DCAS 用户端软件与终端软件平台之间的数据交互。具体实现方式见附录 C。

a)　存储访问接口

DCAS 用户端软件通过调用存储子接口实现对终端存储设备的访问。

b)　网络接口

DCAS 用户端软件通过调用网络子接口实现对网络数据的访问。

c)　过滤接口

DCAS 用户端软件通过调用传输流过滤子接口实现对 ECM、EMM 及 CAT 的接收。

d)　DCAS 管理器

DCAS 用户端软件通过 CA 模块管理子接口实现其向终端软件平台的注册，提供接收终端软件平台解扰请求的机制。

e)　层级密钥接口

DCAS 用户端软件通过调用层级密钥子接口实现对终端安全芯片的认证，以及将加密的密钥载入终端安全芯片完成对码流的解扰。

f)　可分离安全设备接口

DCAS 用户端软件通过可分离安全设备子接口实现对智能卡等可分离外部安全设备的访问。

g)　应用间通信接口

DCAS 用户端软件通过此接口实现同其他 CA 相关应用的数据交换。

8.3 DCAS 用户端软件下载及启动

DCAS 用户端软件下载及启动包括：

a) 终端软件平台下载及启动

终端软件平台下载和启动机制见附录 A。

b) DCAS 用户端软件下载及启动

DCAS 用户端软件下载和启动的机制参见附录 D。

8.4 典型的终端发放流程

典型的 DCAS 终端生产发放使用和软件签名认证流程参见附录 E。

9 终端安全芯片

9.1 安全芯片工作流程

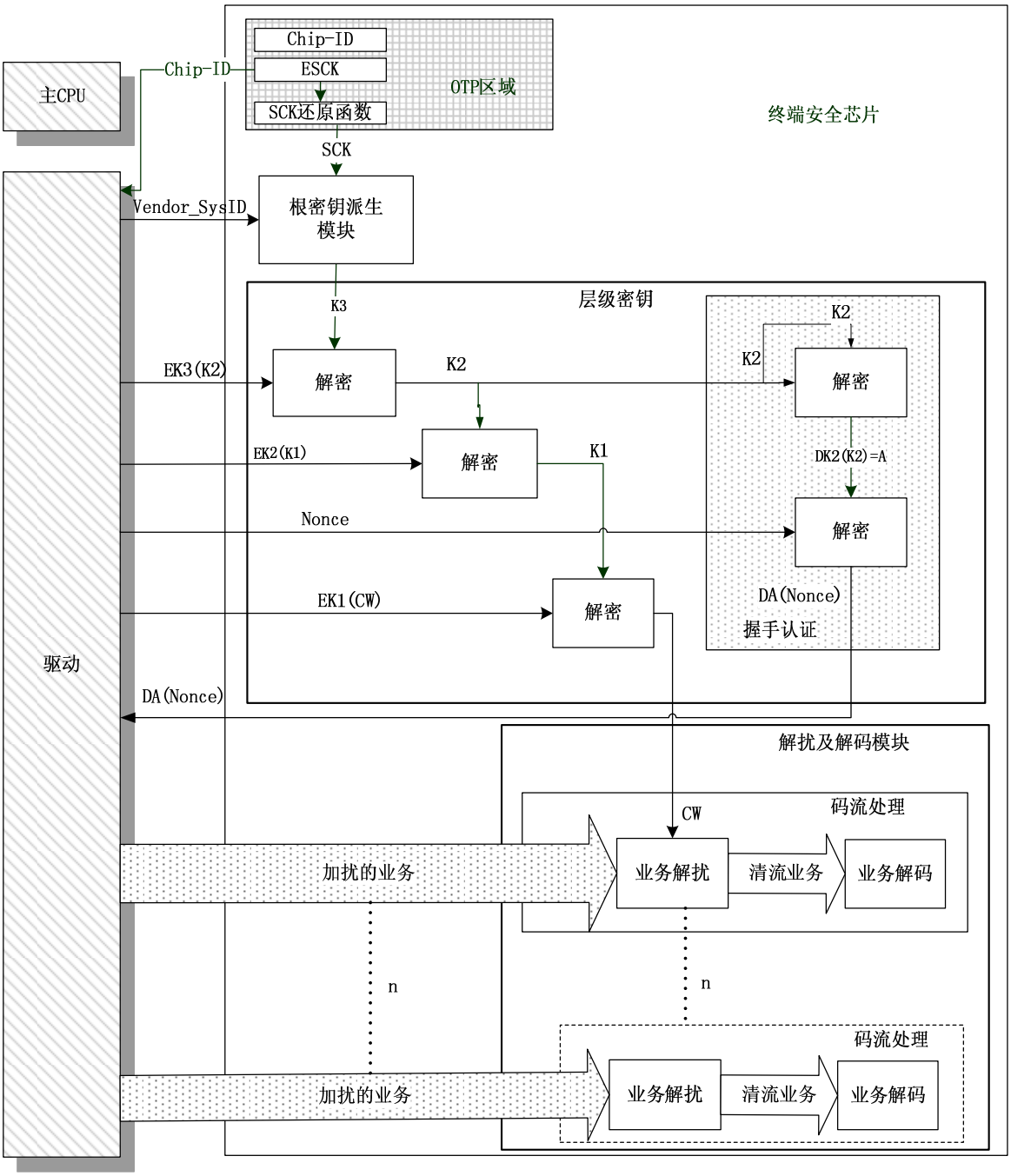


图8　终端安全芯片功能框图

终端安全芯片功能见图 8，终端安全芯片包含：OTP、根密钥派生、层级密钥和解扰及解码等模块。

终端安全芯片通过根密钥派生模块生成根密钥 K3，通过层级密钥模块保障控制字和其它密钥的安全传输及终端安全芯片的合法性。

终端安全芯片的功能不能由主 CPU 实现。

终端安全芯片工作流程：芯片上电后，OTP 通过内置的 ESCK 及 SCK 还原函数生成 SCK,并提供给根密钥派生模块，用于生成根密钥 K3。层级密钥模块接收根密钥 K3 用于相关密钥的解密及握手认证流程。层级密钥模块包含两个方面的功能：对输入的加密密钥实现分层解密；处理握手信息（nonce），并生成认证响应（DA（nonce））。最终解密获得的 CW 被送至解扰及解码模块进行业务解扰和解码。

采用终端安全芯片对本地内容的保护方法见附录 F。

9.2 根密钥派生模块

根密钥派生模块由一组硬件逻辑模块构成，通过其中嵌入的派生机制，并结合输入参数来实现根密钥的派生。任何一家条件接收提供商均通过此派生机制生成不同的根密钥。此方式规避了嵌入单一根密钥带来的安全风险。

根密钥派生模块功能见图 9，根密钥派生模块包括：SCK 初步处理函数、厂商分离函数和根密钥最终派生函数。根据外部输入的 SCK 和 Vendor_SysID，派生出某一条件接收提供商的根密钥。

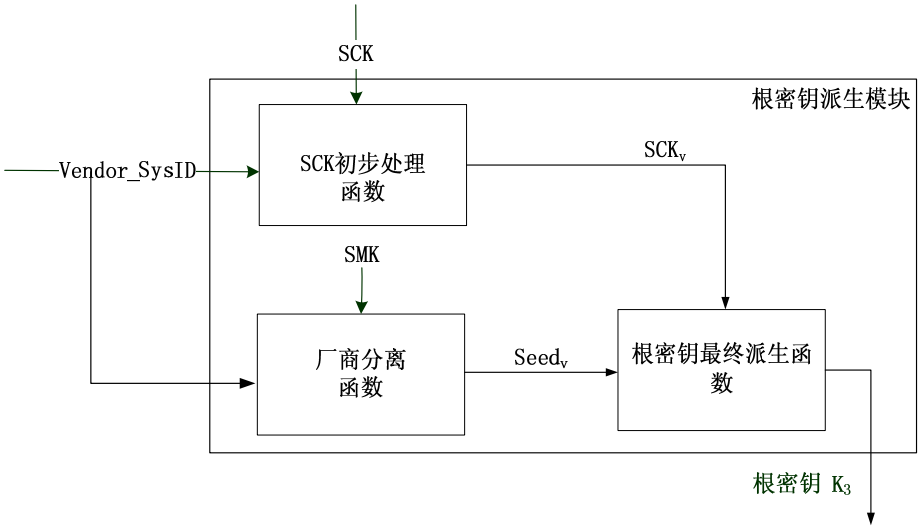


图9 根密钥派生模块功能框图

SCK 初步处理函数模块基于 SCK 和输入的 Vendor_SysID，生成 SCKv。该函数中应使用具有一定强度的密码算法，此强度应保证在已知 Vendor_SysID 和 SCKv 的情况下，SCK 不能够被还原。SCK 初步处理函数应由终端安全芯片厂商提供并由安全数据管理平台认证。

厂商分离函数使用 Vendor_SysID 和 SMK 作为输入值，生成 Seedv。该函数中应使用具有一定强度的密码算法，此强度应保证在已知 Vendor_SysID 和 Seedv 的情况下，SMK 不能够被还原。厂商分离函数应由终端安全芯片厂商提供并由安全数据管理平台认证。

根密钥最终派生函数依据 SCKv 和 Seedv 的输入值派生出根密钥 K3。在实施过程中，根密钥最终派生函数应采用单向函数，使得当知道 K3 和任一输入参数时，不能还原另一个输入参数。例如，当知道 K3 和 SCKv 时，不能还原 Seedv。根密钥最终派生函数应由终端安全芯片厂商提供并由安全数据管理平台认证。

- Vendor_SysID：用于标识条件接收系统，由安全数据管理平台分配，长度为 2 字节
- SCKv：16 字节
- SMK：16 字节，芯片厂商提供的门级数据
- Seedv：16 字节

终端安全芯片可同时支持多种根密钥最终派生函数。根密钥派生模块相关要求参见附录 G。

9.3 层级密钥

9.3.1 三层密钥机制

层级密钥模块功能框图见图10。

本标准规定的终端安全芯片应支持三层密钥机制，更多层级的密钥机制，其安全要求和技术细节不在本标准的描述范围内。

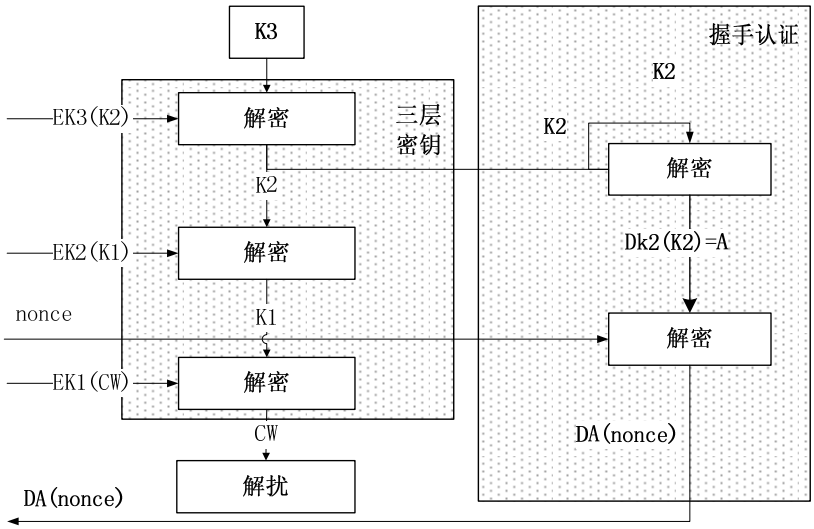


图10　层级密钥模块功能框图

三层密钥机制确保控制字在终端的传输安全。

三层密钥机制通过从根密钥派生模块获得的根密钥 K3，依次解密 EK3(K2)、EK2(K1)、EK1(CW)获得解扰所需的控制字；同时 K2 配合前端发送的握手信息（nonce）成响应与前端完成握手认证。

终端安全芯片应遵循以下流程解密加扰业务：

a)　应接收密文 EK3(K2)，使用 K3 解密该密文，并生成 K2；

b)　应接收密文 EK2(K1)，使用 K2 解密该密文，并生成 K1；

c)　应接收密文 EK1(CW)，使用 K1 解密该密文，并生成 CW；

d)　CW 用于解密加扰业务。

EK3(K2)表示用密钥 K3 加密的数据 K2。

EK2(K1)表示用密钥 K2 加密的数据 K1。

EK1(CW/Key)表示用密钥 K1 加密的数据 CW。

K3 是派生根密钥，长度 16 字节。

K2 是用于解密 K1 的密钥，长度为 16 字节。

K1 是用于解密 CW 的密钥，长度为 16 字节。

CW 是用于解扰业务的密钥，长度为 8 或 16 字节。

层级密钥算法：

a)　在使用 TDES 算法时，密钥每 7 比特后增加 1 比特冗余位，将 112 比特密钥补齐为 128 比特（16 字节）。

b)　层级密钥中的 AES 是指 FIPS PUB 197 中定义的标准 AES-128 算法，计算使用 128 比特，模式为 ECB。

9.3.2　握手认证

终端安全芯片应支持握手认证机制，DCAS 前端实现对终端安全芯片的合法性进行验证，确保授权下发到合法的终端。同时，握手认证机制也可以被用于其它一些安全功能。

握手认证机制应遵循以下流程：

a)　终端安全芯片应通过 API 驱动接口接收 Vendor_SysID、EK3(K2)和 Nonce。

b)　终端安全芯片利用派生的 K3 解密 EK3（K2）得到 K2。

c)　终端安全芯片应通过 K2 解密 K2 生成 DK2(K2),记为 A。

d) 终端安全芯片应使用 A 解密 nonce，生成 DA(nonce)。

e) 终端安全芯片可向前端提出 DA(nonce)。

f) 前端对 DA（nonce）做出响应。

DK2(K2)表示用用密钥 K2 解密 K2 的过程。

A 表示 DK2(K2)的结果，长度为 16 个字节。

nonce 握手认证信息，长度为 16 字节。

DA(nonce)表示用 A 解密随机数 Nonce 得到的结果。

DCAS 前端与终端安全芯片的握手认证流程见附录 H。

9.4 OTP 区域

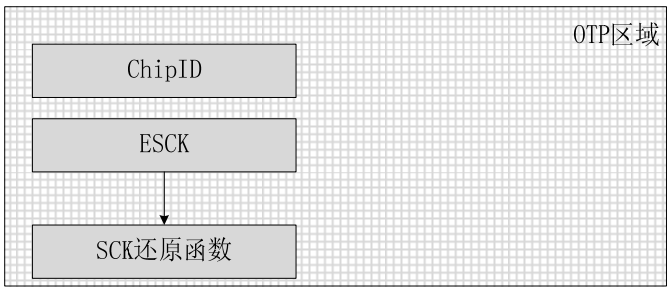


图11 OTP 区域功能框图

OTP 区域功能框图见图 11，用于存储 ChipID、ESCK 和 SCK 还原函数等信息。SCK 还原函数逻辑电路读取 OTP 区域中的 ESCK，将 ESCK 还原成 SCK，并将 SCK 提供给根密钥派生模块。

SCK 是派生根密钥所需的必要安全信息，SCK 不能直接存储在 OTP 区域中。

a) SCK 是终端安全芯片密钥，每芯片唯一，长度为 16 字节，由安全数据管理平台生成。

b) ESCK 是加密后的 SCK，由安全数据管理平台提供，存储于 OTP 区域，长度等于 SCK 长度。

c) ChipID 是终端安全芯片的公开标识符，包括了表明芯片厂商和型号的信息，以及芯片的一个全球唯一标识符，长度 8 字节，由安全数据管理平台分配，数据格式见表 1。

表1 ChipID 数据格式

	长度	数据类型
芯片厂商 ID（Chip Manufacturer ID）	8 比特	uimsbf
芯片类型（Chip Type）	12 比特	uimsbf
保留（Reserved）	12 比特	uimsbf
芯片标识	32 比特	uimsbf

芯片厂商 ID：芯片厂商的唯一标识，8 比特。

芯片类型：某一芯片厂商生产的芯片型号标识，由安全数据管理平台分配，12 比特。

保留位：12 比特。

芯片标识：某一芯片制造厂商生产的某一芯片类型芯片的序列标识，32 比特。

附　录　A
（规范性附录）
终端启动加载软件启动机制

DCAS 用户端软件安全机制基于一种自底向上的信任链，从终端安全芯片至启动加载软件、终端软件平台和 DCAS 用户端软件，采用数字签名技术建立信任链，只有在在信任链中的每个环节都通过签名校验后，信任链的后一环节方可启动。

A.1　信任链基本原则

在整个信任链机制中：

a)　每个软件组件都应被签名；

b)　签名应被添加到被签名的软件组件中；

c)　每个软件组件应在验证其后装入的软件组件后，才移交控制权；

d)　验证密钥应在实施验证的软件组件中，并且也被验证过；

e)　用于验证初始的软件组件的密钥应预置在终端主芯片中，并应有一些 OTP 比特进行标识；

f)　下载的软件组件应被签名，并被验证后接收。

终端软件执行过程中应始终遵循信任链机制。

终端启动加载软件需要防止非法软件被终端加载，下载及运行。

CPU 应执行一定的安全代码，检查在 CPU 外部 Flash 中已签名的启动加载软件。启动加载软件则需要验证已签名的主终端软件。

A.2　启动签名验证

启动加载软件验证过程见图A.1。

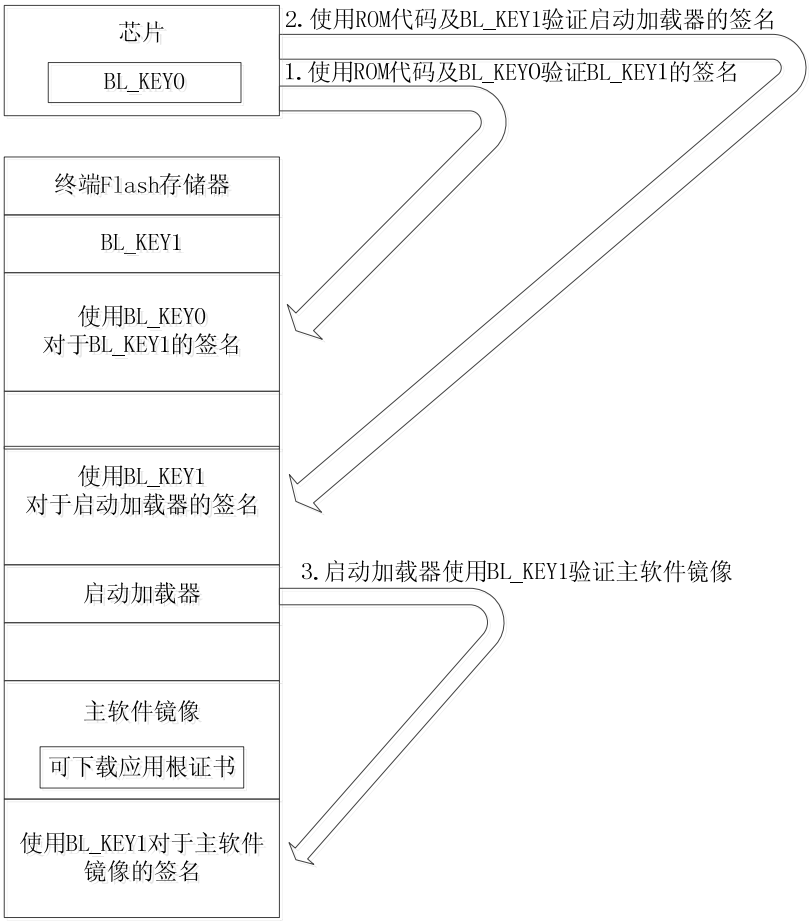


图 A.1　启动加载软件验证过程

终端安全芯片应在芯片安全区域中嵌入由安全数据管理平台分配的公钥，这里记做 BL_KEY0。安全数据管理平台在任何情况下不应泄露 BL_KEY0 的私钥。图 A.1 中的 BL_KEY0 指其公钥。

终端安全芯片在启动时应执行其安全区域中的代码从终端读取额外的公钥，这里记做 BL_KEY1，以及 BL_KEY1 的签名，并应使用 BL_KEY0 验证 BL_KEY1 的签名是否正确。图 A.1 中的 BL_KEY1 指其公钥。

在成功验证 BL_KEY1 后，终端安全芯片应使用 BL_KEY1 对启动加载软件进行验证。

启动加载软件应使用 BL_KEY1 对终端主软件的签名进行验证。

可下载的应用应使用从根证书开始的证书链并遵循本标准中的应用下载，启动的签名机制。由终端主软件完成对可下载的应用签名验证。

上述过程提到的所有签名都应使用至少 2048 比特 RSA 密钥，SHA-1 或 SHA-256(建议使用 SHA-256)Hash 算法，密钥存储格式建议使用 RFC3447。

A.3　密钥管理

启动加载软件密钥说明见表 A.1。

表 A.1　启动加载软件密钥说明

密钥名称	密钥属主	被签名	用于签名	备注
BL_KEY0	安全数据管理平台	N/A	BL_KEY1	公钥被嵌入芯片
BL_KEY1	安全数据管理平台（密钥托管）	BL_KEY0	启动加载软件 主软件镜像	

对于 BL_KEY0：

安全数据管理平台应管理其内部 BL_KEY0 的数据库，应负责分发 BL_KEY0 至芯片厂商，用于嵌入指定的芯片组。

对于 BL_KEY1：

BL_KEY1 所有者是安全数据管理平台。无论上述哪种 BL_KEY1 持有情形，为了保证终端软件可以被 BL_KEY1 正确签名，BL_KEY1 所有者应当向安全数据管理平台进行密钥托管。

安全数据管理平台应提供使用 BL_KEY0 签名 BL_KEY1 的方法和过程。BL_KEY1 所有者应向安全数据管理平台提供终端的具体信息，这些信息包括，芯片型号，BL_KEY1 的公私钥。如果安全数据管理平台批准该终端设备，则应将 BL_KEY1 的公钥使用 BL_KEY0 的私钥签名，并将签名结果连同 BL_KEY1 的公钥一并返还给 BL_KEY1 所有者，用于将 BL_KEY1 的公钥及签名结果置入终端设备。

A.4　启动加载软件安全要求

启动加载软件安全要求主要针对启动过程和下载过程两个方面。芯片应支持从 RAM 中启动功能（BFR）。应满足信任链机制，所有由启动加载软件装入，启动，下载，更新的软件组件应被签名和验证。启动加载软件只负责验证平台软件，由平台软件下载的应用软件应由平台软件验证。

对于启动过程：

启动加载软件应在成功验证后续软件组件签名后，开始执行其代码。启动加载软件应在每一次后续软件组件重启后重新验证其签名。只有存储在终端 Flash 中的软件组件可以被执行。所有软件组件的签名验证和执行应在 RAM 中进行。验证后的软件组件，如有需要可以进行解压缩然后运行。

对于下载过程：

启动加载软件只有在 RAM 中验证下载的软件镜像签名正确后，方可将所下载的软件及其签名写入 Flash。软件下载成功后，终端应立即执行一次完全重启。如果所下载的软件超过启动加载软件分配的 Flash 最大尺寸，启动加载软件应拒绝该软件，并执行重启。应避免降级可升级软件组件。

A.5　启动加载软件和安全芯片的性能要求

为了保证终端的用户体验，应采用安全芯片为 Hash 算法提供硬件加速。

附　录　B
（资料性附录）
DCAS 用户端软件签名、验签过程

B.1　签名过程

B.1.1　签名对象

DCAS 用户端软件的签名对象是终端软件包。

终端软件数据包应打包成 zip 格式，其中 zip 文件中仅包含 DCAS 用户端软件一个文件，以 jar 包形式提供。这个 jar 文件为签名对象。终端需要先从 zip 文件中将 jar 文件解压释放，然后进行签名验证。

B.1.2　算法支持

Hash 算法支持 SHA1,SHA256；

签名算法采用 RSA

定义：

HASH_SHA1	采用 SHA1Hash 算法
HASH_SHA256	采用 SHA256Hash 算法
SIGN_SHA1_RSA	采用 SHA1 摘要，RSA 签名
SIGN_SHA256_RSA	采用 SHA256 摘要，RSA 签名

B.1.3　签名文件

签名文件中包含对软件的数字签名，其格式遵循 ASN.1 DER 结构。

```
Signature ::= SEQUENCE {
certificateIdentifier                    AuthorityKeyIdentifier,
hashSignatureAlgorithm                   OBJECT IDENTIFIER,
signatureValue                           BIT STRING }
```

certificateIdentifier: 验证该签名的证书标识：

```
AuthorityKeyIdentifier ::= SEQUENCE {
keyIdentifier                    [0] KeyIdentifier OPTIONAL,
authorityCertIssuer              [1] GeneralNames, OPTIONAL,
authorityCertSerialNumber        [2]          CertificateSerialNumber
OPTIONAL }
```

AuthorityKeyIdentifier 中由 X.509 规范定义，在本方案中必须包含 authorityCertIssuer 和 authorityCertSerialNumber 字段。

hashSignatureAlgorithm: 指示签名所使用的 Hash 算法，本方案规定支持 SHA-1 和 SHA-256：

```
sha-1 OBJECT IDENTIFIER ::=
{iso(1) identifier-organization(3) oiw(14) secsig(3) algorithm(2) 26}
```

```
    sha-256 OBJECT IDENTIFIER::=
    {joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) 1 }
```

signatureValue: 签名位串。

B.1.4 证书数据

B.1.4.1 描述

证书提供一个公钥，通过与终端计算的软件包 Hash 值、签名文件中包含的签名值进行运算，来验证软件包的合法性。证书本身被更高级证书机构所签名，这样就形成了一个证书链，证书链的顶端是根证书。根证书是一个自签名的证书，即其合法性由其自身所保证。根证书必须由统一的证书中心颁发。

证书链包含在一个证书文件中，和应用程序包及应用签名文件一同下发。

B.1.4.2 格式

证书文件格式见表 B.1。

表 B.1 证书文件格式

语法	位宽	格式
CertificateFile() {		
certificate_count	16	uimsbf
for (i=0; i<certificate_count; i++) {		
certificate_length	24	uimsbf
certificate()		
}		
}		

certificate_count：存储证书文件中证书的个数

certificate_length：表示证书所占字节长度

certificate()：包含 X.509 格式的证书数据。

对于 DCAS 终端软件的证书链，包含了三个证书：安全数据管理平台根证书、运营商证书、DCAS 前端平台证书；这三个证书按照上面的顺序依次存储在证书文件中。即：i=0 时是安全数据管理平台根证书，i=1 时是运营商证书，i=2 时是 DCAS 前端平台证书。

B.2 XML文件

B.2.1 描述

DCAS 终端软件的下发遵循应用软件的自动部署规范。因此有关终端软件包、证书链、签名文件等的信息使用统一的 XML 格式进行记录，基于 XML 语法编码的应用信令 XML Schema 定义见表 B.2。

表 B.2　应用信令 XML Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ngb="urn:ngb:middleware-xait:2010"
targetNamespace="urn:ngb:middleware-xait:2010" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
    <xsd:element name="applicationList" type="ngb:ApplicationList"/>
    <xsd:complexType name="ApplicationList">
        <xsd:sequence>
            <xsd:element name="application" type="ngb:Application" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="networkId" type="xsd:unsignedShort" use="required"/>
        <xsd:attribute name="versionMajor" type="ngb:Hexadecimal8bit" use="required"/>
        <xsd:attribute name="versionMinor" type="ngb:Hexadecimal8bit" use="required"/>
    </xsd:complexType>
    <xsd:complexType name="Application">
        <xsd:sequence>
            <xsd:element name="applicationName" type="ngb:MultilingualType" maxOccurs="unbounded"/>
            <xsd:element name="applicationIdentifier" type="ngb:ApplicationIdentifier"/>
            <xsd:element name="applicationDescriptor" type="ngb:ApplicationDescriptor"/>
            <xsd:element name="applicationSpecificDescriptor"
type="ngb:ApplicationSpecificDescriptor"/>
            <xsd:element name="applicationUsageDescriptor" type="ngb:ApplicationUsageDescriptor"
minOccurs="0"/>
            <xsd:element name="applicationTransport" type="ngb:TransportProtocolDescriptorType"
maxOccurs="unbounded"/>
            <xsd:element name="applicationProvider" type="ngb:MultilingualType" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="applicationInstallDescriptor" type="ngb:ApplicationInstallDescriptor"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ApplicationIdentifier">
        <xsd:sequence>
            <xsd:element name="orgId" type="xsd:unsignedInt"/>
            <xsd:element name="appId" type="xsd:unsignedShort"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ApplicationType">
        <xsd:choice>
            <xsd:element name="dvbApp" type="ngb:DvbApplicationType"/>

                                <xsd:element name="ngbApp" type="ngb:NgbApplicationType"/>
```

表 B.2　应用信令 XML Schema（续）

```
        </xsd:choice>
    </xsd:complexType>
    <xsd:simpleType name="DvbApplicationType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="DVB-J"/>
            <xsd:enumeration value="DVB-HTML"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="NgbApplicationType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="NGB-J-Xlet"/>
            <xsd:enumeration value="NGB-J-MIDlet"/>
            <xsd:enumeration value="NGB-H"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="VisibilityDescriptor">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="NOT_VISIBLE_ALL"/>
            <xsd:enumeration value="NOT_VISIBLE_USERS"/>
            <xsd:enumeration value="VISIBLE_ALL"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="PlatformVersion">
        <xsd:sequence>
            <xsd:element name="platform_profile" type="ngb:Hexadecimal16bit"/>
            <xsd:element name="versionMajor" type="ngb:Hexadecimal8bit"/>
            <xsd:element name="versionMinor" type="ngb:Hexadecimal8bit"/>
            <xsd:element name="versionMicro" type="ngb:Hexadecimal8bit"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="IconDescriptor">
        <xsd:attribute name="filename" type="xsd:string" use="required"/>
        <xsd:attribute name="size" type="xsd:unsignedShort" use="optional"/>
        <xsd:attribute name="aspectRatio" type="ngb:AspectRatio" use="optional"/>
    </xsd:complexType>
    <xsd:simpleType name="AspectRatio">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="4_3"/>
```

表 B.2　应用信令 XML Schema（续）

```
            <xsd:enumeration value="16_9"/>
            <xsd:enumeration value="1_1"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="StorageCapabilities">
        <xsd:sequence minOccurs="0">
            <xsd:element name="storageProperty" type="ngb:StorageType"/>
            <xsd:element name="isStorable" type="xsd:boolean"/>
            <xsd:element name="canCache" type="xsd:boolean"/>
        </xsd:sequence>
        <xsd:attribute name="notlaunchableFromBroadcast" type="xsd:boolean" use="required"/>
        <xsd:attribute name="launchableCompletelyFromCache" type="xsd:boolean" use="optional"/>
        <xsd:attribute name="launchableWithOlderVersion" type="xsd:boolean" use="required"/>
    </xsd:complexType>
    <xsd:simpleType name="StorageType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="STANDALONE"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="ApplicationUsageDescriptor">
        <xsd:sequence>
            <xsd:element name="applicationUsage" type="xsd:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
            <xsd:element name="applicationIntroduction" type="ngb:MultilingualType" minOccurs="0"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TransportProtocolDescriptorType" abstract="true"/>
    <xsd:complexType name="HTTPTransportType">
        <xsd:complexContent>
            <xsd:extension base="ngb:TransportProtocolDescriptorType">
                <xsd:sequence>
                    <xsd:element name="URLBase" type="xsd:anyURI"/>
                    <xsd:element name="URLExtension" type="xsd:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

表 B.2　应用信令 XML Schema（续）

```
<xsd:complexType name="OCTransportType">
    <xsd:complexContent>
        <xsd:extension base="ngb:TransportProtocolDescriptorType">
            <xsd:sequence>
                <xsd:choice minOccurs="0">
                    <xsd:element name="ngbTriplet" type="ngb:NGBTriplet"/>
                    <xsd:element name="textualId" type="ngb:TextualIdentifier"/>
                </xsd:choice>
                <xsd:element name="componentTag" type="ngb:ComponentTagType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ApplicationSpecificDescriptor">
    <xsd:choice>
        <xsd:element name="ngbjDescriptor" type="ngb:NGBJDescriptor"/>
        <xsd:element name="ngbhDescriptor" type="ngb:NGBHDescriptor"/>
    </xsd:choice>
</xsd:complexType>
<xsd:complexType name="NGBJDescriptor">
    <xsd:sequence>
        <xsd:element name="parameter" type="xsd:string"/>
        <xsd:element name="ngbjApplicationStructure" type="ngb:NGBJApplicationStructure"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="NGBHDescriptor">
    <xsd:sequence>
        <xsd:element name="parameter" type="xsd:string"/>
        <xsd:element name="ngbhApplicationStructure" type="ngb:NGBHApplicationStructure"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="NGBJApplicationStructure">
    <xsd:sequence>
        <xsd:element name="baseDirectory" type="xsd:string"/>
        <xsd:element name="classPathExtension" type="xsd:string"/>
        <xsd:element name="initialClass" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="NGBHApplicationStructure">
    <xsd:sequence>
```

表 B.2　应用信令 XML Schema（续）

```
            <xsd:element name="physicalRoot" type="xsd:string"/>
            <xsd:element name="initialPath" type="xsd:string"/>
            <xsd:element name="boundaryExtension" type="xsd:anyURI" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="MultilingualType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="lang" type="ngb:ISO639-2" use="required"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="NGBTriplet">
        <xsd:attribute name="OrigNetId" type="ngb:OrigNetId" use="required"/>
        <xsd:attribute name="TSId" type="ngb:TSId" use="required"/>
        <xsd:attribute name="ServiceId" type="ngb:ServiceId" use="required"/>
    </xsd:complexType>
    <xsd:complexType name="TextualIdentifier">
        <xsd:attribute name="DomainName" type="ngb:DomainType" use="optional"/>
        <xsd:attribute name="ServiceName" type="ngb:Service" use="required"/>
    </xsd:complexType>
    <xsd:complexType name="ComponentTagType">
        <xsd:attribute name="componentTag" type="ngb:Hexadecimal8bit"/>
    </xsd:complexType>
    <xsd:complexType name="ApplicationDescriptor">
        <xsd:sequence>
            <xsd:element name="type" type="ngb:ApplicationType"/>
            <xsd:element name="visibility" type="ngb:VisibilityDescriptor" minOccurs="0"/>
            <xsd:element name="priority" type="ngb:Hexadecimal8bit"/>
            <xsd:element name="version" type="ngb:Version"/>
            <xsd:element name="platformVersion" type="ngb:PlatformVersion" maxOccurs="unbounded"/>
            <xsd:element name="icon" type="ngb:IconDescriptor" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="storageCapabilities" type="ngb:StorageCapabilities" minOccurs="0"/>
            <xsd:element name="size" type="xsd:unsignedInt"/>
            <xsd:element name="bootPath" type="xsd:string" minOccurs="0"/>
            <xsd:element name="classPath" type="xsd:string" minOccurs="0"/>
            <xsd:element name="needPrevLoad" type="xsd:boolean" default="false" minOccurs="0"/>
            <xsd:element name="classPrevLoad" type="xsd:string" minOccurs="0"/>
            <xsd:element name="needPrevInit" type="xsd:boolean" default="false" minOccurs="0"/>
            <xsd:element name="requiredApps" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="exportPackages" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
```

表 B.2　应用信令 XML Schema（续）

```
            <xsd:element name="importPackages" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ApplicationInstallDescriptor">
        <xsd:sequence>
            <xsd:element name="powerOnRunType" type="xsd:unsignedByte"/>
            <xsd:element name="autoRunOrder" type="xsd:unsignedByte"/>
            <xsd:element name="upgradeType" type="xsd:unsignedByte"/>
            <xsd:element name="downloadRunType" type="xsd:unsignedByte"/>
            <xsd:element name="publishTime" type="xsd:string"/>
            <xsd:element name="runningType" type="xsd:unsignedByte"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="Hexadecimal8bit">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[0-9 a-f A-F]{1,2}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="Hexadecimal16bit">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[0-9 a-f A-F]{1,4}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="ISO639-2">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\c\c\c"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="Version">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="[0-9 a-f A-F]{2}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="DescriptionLocation">
        <xsd:restriction base="xsd:anyURI"/>
    </xsd:simpleType>
    <xsd:simpleType name="OrigNetId">
        <xsd:restriction base="xsd:unsignedShort"/>
    </xsd:simpleType>
    <xsd:simpleType name="TSId">
        <xsd:restriction base="xsd:unsignedShort"/>
```

表 B.2　应用信令 XML Schema（续）

```
    </xsd:simpleType>
    <xsd:simpleType name="ServiceId">
        <xsd:restriction base="xsd:unsignedShort"/>
    </xsd:simpleType>
    <xsd:simpleType name="DomainType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="((.|\n|\r)*)?(\.(.|\n|\r)*)+"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="Service">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="(.|\n|\r)+"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

由于 DCAS 终端软件需要进行签名、验签操作，以保证软件包的合法性， DCAS 终端软件的 XML 信息需要注意以下几点：

a)　applicationIdentifier 子元素 appId 值的范围和分配

appId 是一个无符号短整数类型(16bit)，描述的是应用软件的编号，对应用软件编号的分配策略见表 E.2，DCAS 终端软件的编号范围应在 0x4000—0x7fff。

表 B.3　应用软件编号分配策略

applicationIdentifier	应用类型
0x0000	不能被使用
0x0001—0x3fff	分配给没有签名的应用
0x4000—0x7fff	分配给有签名的应用
0x8000—0x9fff	分配给特权应用
0xa000—0xfffd	保留
0xfffe	通配符，表示同一个组织的签名应用
0xffff	通配符，表示同一个组织的所有应用

b)　application 元素的 applicationTransport 子元素

DCAS 终端软件的下载采用 http 传输协议（HTTPTransportType）。其包含两个子元素：

URLBase：定义多个下载文件的 URL 的共享部分。

URLExtension：定义了多个下载文件的不同部分，具体如下：

终端软件包：dcas_application_<version>.zip

证书链：ngb.certChain.1

签名文件：ngb.signature.<appname>.1

上述三个文件在 URLExtention 元素中以空格分隔，下载时的路径分别为

<URLBase>/dcas_application_<version>.zip

<URLBase>/ngb.certChain.1

<URLBase>/ngb.signature.<appname>.1

DCAS 前端按 XML 格式生成包含一个 application 元素的 xml 文件，提交给应用自动部署服务器，服务器将其添加到统一的应用程序描述列表中，供前端下载。

B.3 验签过程

终端下载 XML 后，从 DCAS 终端软件对应的 application 元素中解析出软件包下载地址，证书链文件下载地址，签名文件下载地址，从服务器分别下载以上文件。

B.3.1 证书链校验

终端根据证书链文件的格式解析证书链，并将根证书与本地预埋根证书做对比，在保证根证书合法的情况下，验证证书链的合法性。

若证书链验证失败，则取消本次软件更新。

B.3.2 终端软件验签

终端根据签名文件的格式，解析出签名算法标识、签名值，根据签名算法标识重新计算软件包的哈希值，并将该哈希值连同证书公钥、签名值作为输入，调用验签相关接口进行合法性校验。

若签名验证失败，则取消本次软件更新；若验证通过，则交由中间件自动部署管理器进行软件部署。

附　录　C
（规范性附录）
DCAS 应用程序接口

标准的 Java 虚拟机方案已在业界被广泛用于下载和启动应用。在此之上，增加对于 DCAS 基本实现的支持，可提供 DCAS 用户端应用的运行环境，见图 C.1。

DCAS 用户端软件是一个运行于支持 Java 运行环境终端软件平台上的 Xlet 应用;运行已下载的 DCAS 用户端软件，可实现对业务解扰。

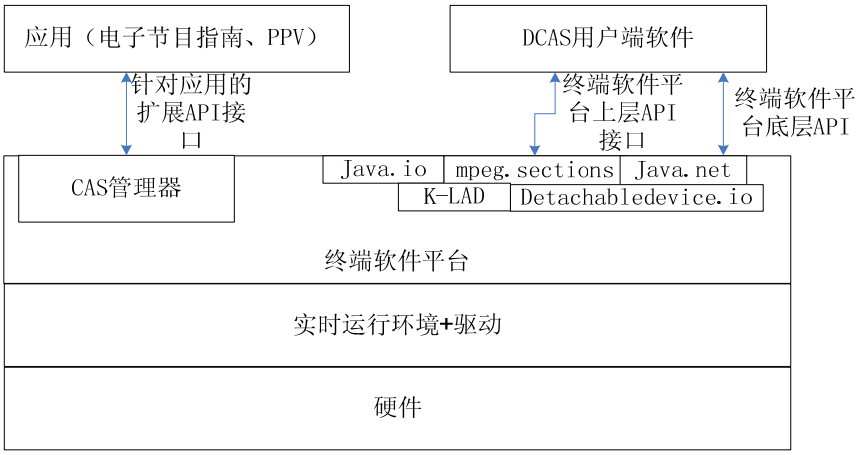


图 C.1　DCAS 应用程序接口示意图

C.1　应用程序接口类型

C.1.1　应用扩展API

通过扩展 DCAS 应用 API，终端软件平台上的 CAS 管理模块可以同 Java 应用进行最基本的 CA 信息传输，而不受在 Java 应用及 DCAS 应用之间使用 IXC 的限制。

DCAS 用户端软件需要终端软件平台为应用实现 DCAS 扩展 API。

C.1.2　终端软件平台上层API

终端软件平台上层 API 定义了 CA 模块管理器，来实现管理针对业务解扰的请求（也就是解扰视音频流）。一个 DCAS 用户端软件应用必须将 CA 模块在 CA 模块管理器中注册，用来接收网络中来自终端设备终端软件平台发出的解扰请求。

DCAS 用户端软件需要终端软件平台实现 DCAS 终端软件平台上层 API。

C.1.3　终端软件平台底层API

DCAS 用户端软件需要终端软件平台实现以下类型的 API 集，其中除了现有 Java API，还包括对于终端安全芯片访问所需的扩展 API。

C.1.3.1　网络API

DCAS 用户端软件可以使用 Java 网络 API 来访问网络资源，比如与前端 CA 服务器的互联。

DCAS 用户端软件需要终端软件平台根据 Java.net 中的定义实现现有 Java 网络 API。

C.1.3.2　MPEG Section过滤API

DCAS 用户端软件使用 MPEG Section 过滤 API 来载入用于 CA 的 MPEG Section。CA 相关的数据可包括 ECM，EMM 及 CAT 表。

DCAS 用户端软件需要终端软件平台根据 org.davic.mpeg.sections，org.davic.mpeg.TransportStream 和 org.davic.net.tuning.NetworkInterface 中的定义实现 MPEG Section 过滤 API。

C.1.3.3 终端安全芯片及解扰器API

终端安全芯片及解扰器 API 扩展用于标准的终端安全芯片。

终端安全芯片为 DCAS 用户端软件提供标准的方式来认证终端设备的硬件或芯片。终端安全芯片 API 为 DCAS 用户端软件应用提供将密钥载入终端安全芯片的机制来安全地提供网络控制字给终端设备的解扰模块。

DCAS 用户端软件需要终端软件平台部署终端安全芯片及解扰 API。

C.1.3.4 永久存储API

DCAS 用户端软件可以使用现有的 Java API 来访问终端存储，包括将数据保存在非易失性存储器中。

DCAS 用户端软件需要终端软件平台根据 java.io 的定义来实现永久存储 API。

DCAS 用户端软件可以在永久存储的文件系统中通过使用特定的目录来保存数据。终端软件平台需要提供适当的函数来读取该存储文件系统的根目录名称。

C.1.4 其他API

DCAS 用户端软件核心同时需要终端软件平台实现以下类型的 API。

DCAS 用户端软件核心可以使用 ClassLoader API 在实时运行环境中装载额外的执行对象。

在启动了安全策略的平台上，DCAS 应用必须被授予一个 java.lang.RuntimePermission(“createClassLoader”)。

C.2 应用接口调用时序

本条提供了使用 DCAS API 的两种情况：CASModule 注册和频道选择切换，CASModule 注册见图 C.2，频道选择切换见图 C.3。

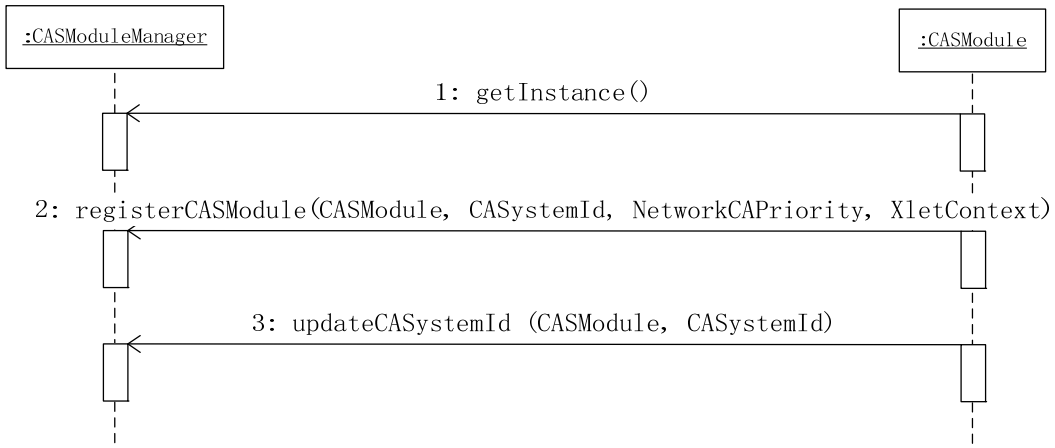


图 C.2 CA 模块在 CASModuleManager 中的注册

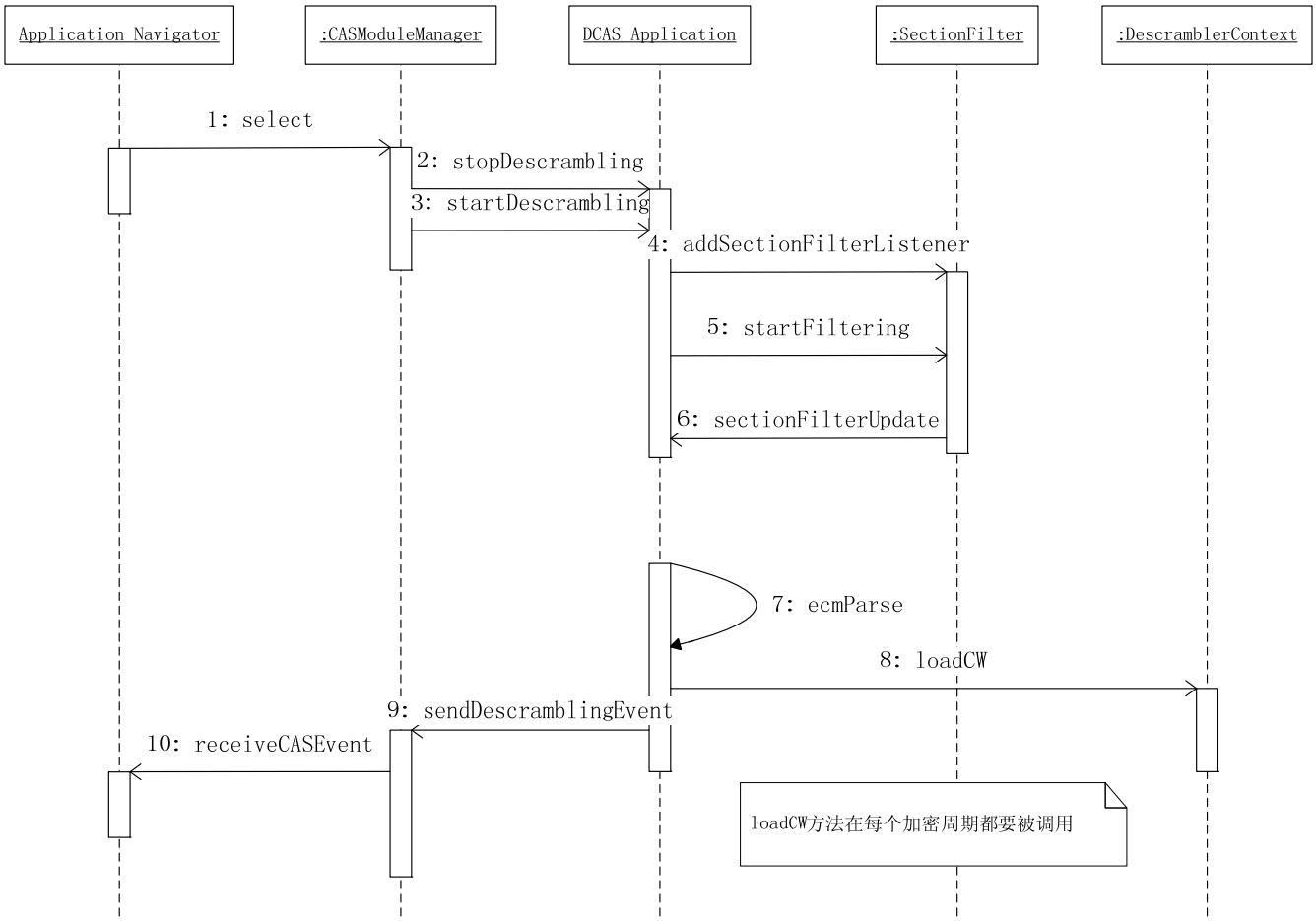


图 C.3　频道选择切换

C.3　应用程序接口描述

应用程序接口名称见表 C.1。

表 C.1　应用程序接口

API 名称	包名称
终端软件平台上层 API	org.ngb.net.cas.module
应用扩展 API	org.ngb.net.cas.event
终端软件平台底层 API	org.ngb.net.cas.controller
网络 API	java.net
Section 过滤 API	org.davic.mpeg.sections
永久性存储 API	java.io
Class Loader API	java.lang

C.3.1　终端软件平台上层API包

C.3.1.1　CASModule接口

本接口描述用于表示请求解扰一组基本流的 CASModule 对象。

C.3.1.1.1 方法

C.3.1.1.1.1 StartDescrambling

语法：public void startDescrambling(CASSession casSession,
CAServiceComponentInfo[] casci)

描述：本方法由终端软件平台调用，请求 CASModule 解扰给定会话中的一组基本流。

DCAS 应用从 CAS 会话中可以得到相关 NetworkInterface 对象。

从 NetworkInterface 中，可以获取当前 TransportStream 对象，用于 org.davic.mpeg.sectionsAPI 进行 ECM section 过滤。

输入：

Cassession:请求解扰操作的会话

casci：CA 业务组件信息数组。该数组可以用于获取相关 ECM PID,以及用以 DCAS 装载控制字的 DescramblerContext 对象

输出：

无

C.3.1.1.1.2 updateDescrambling

语法：public void updateDescrambling(CASSession casSession,
CAServiceComponentInfo[] casci)

描述：终端软件平台调用本方法来更新 CASModule 中的解扰组件列表。

根据请求，CASModule 将开始解扰添加到数组的组件，并停止解扰移除的组件。

对于更新后不变的成分，不发生任何变化。

注：本方法很少被调用，通常在一个会话中由于 PMT 变化而发生。

终端软件平台还可在 CAS 会话如会话类型发生变化时通过调用本方法通知 CASModule。

输入：

casSession：请求解扰操作的会话

casci：CA 业务组件信息数组，该数组可以用于获取相关 ECM PID,以及用以 DCAS 装载控制字的 DescramblerContext 对象

输出：

无

C.3.1.1.1.3 stopDescrambling

语法：public void stopDescrambling(CASSession casSession)

描述：终端软件平台调用本接口请求 CASModule 停止解扰给定会话中的所有组件。

输入：

casSession-请求解扰操作的会话

输出：

无

C.3.1.2 CASModuleManager类

用来注册所有被 DCAS 应用实现的 CASModule。

C.3.1.2.1 方法

C.3.1.2.1.1 getInstance

语法：public static CASModuleManager getInstance()
throws java.lang.SecurityException

描述：本方法用于获取一个 CASModuleManager 单例。

输入：

无

输出：

CASModuleManager 实例

异常处理：

java.lang.SecurityException-如安全策略被强制启用，但调用方没有被赋予一个 org.ngb.net.ca.module.CASPermission。

C.3.1.2.1.2 RegisterCASmodule

语法：public void registerCASModule(CASModule aModule,
int caSystemId,
int networkCAPriority,
java.lang.Object xletContext)
throws java.lang.IllegalArgumentException

描述：本方法用于 DCAS 应用在终端软件平台上注册一个 CASModule。

输入：

aModule:需要注册的 CASModule

caSystemId：CASModule 管理的 caSystemId

networkCAPriority：用于在超过一个 CASModule 在 CASModuleManager 中注册时使用，运营商可根据每个 CASModule 决定是否该参数可选，当优先级策略启用时，为运营商需要为每个 CASModule 指定优先级。终端软件平台应选择已注册，具有最高优先级，并且所管理的 caSystemId 在 PMT 中有相应 CA 描述符的 CASModule 发出解扰请求。当优先级策略禁用时，DCAS 应用应给该参数置零，CASModule 的决定方法由终端软件平台自行实现

xletContext：希望进行注册 CASModule 的 DCAS 应用的 XletContext，用于终端软件平台决定 DCAS 应用的身份

输出：

无

异常处理：

java.lang.IllegalArgumentException-如果指定的 CASModule 实例已被注册。

C.3.1.2.1.3 RegisterCASmodule

语法：public void updateCASystemId(CASModule aModule, int caSystemId)
throws java.lang.IllegalArgumentException

描述：本方法用于 DCAS 应用向应用软件平台更新某 CASModule 中的 CASystemId。

输入：

aModule-指定 CASModule

caSystemId -module 管理的新 caSystemId

输出:

无

异常处理：

java.lang.IllegalArgumentException 如果给定的 CASModule 实例尚未注册。

C.3.1.2.1.4 SendDescramblingEvent

语法：public void sendDescramblingEvent(CASModule aModule,
CASSession casSession,
CAStatus aCAStatus)
throws java.lang.IllegalArgumentException

描述：本方法被用于DCAS应用向终端软件平台返回一个CAStatus当某个DescramblerContext每次由于相应服务中的加扰成分的加扰组件改变而变化时，DCAS应用必须发送CAStatus用于指示解扰是否成功。如果任何一个组件解扰失败，CAStatus必须通知此服务的解扰失败。

当终端软件平台收到一个新的CAStatus时，应继续通过在扩展API中定义的CAS Event继续将该信息传至相应应用。

输入：

aModule：指定CASModule

casSession：解扰请求操作的会话

aCAStatus：需要发送的CAStatus

输出：

无

异常处理：

java.lang.IllegalArgumentException-如果所给的CASModule实例没有被注册。

C.3.1.2.1.5 unregisterCASModule

语法：public void unregisterCASModule(CASModule aModule)
throws java.lang.IllegalArgumentException

描述：本方法用于DCAS应用从终端软件平台上取消某CASModule的注册。

输入：

aModule-需要取消注册的CASModule

输出：

无

异常处理：

java.lang.IllegalArgumentException 如果给定的CASModule尚未注册。

C.3.1.2.1.6 getChipControllers

语法：public ChipController[] getChipControllers()

描述：本方法用于DCAS应用从终端软件平台请求可被使用的芯片控制器列表，本方法为每个终端安全芯片返回一个芯片控制器。很多终端只支持单个芯片控制器，这种情况下，返回的数组中只含有一个元素。

输入：

无

输出：

一个芯片控制器数组

C.3.1.2.1.7 SetcurrentController

语法：public void setCurrentController(CASModule aModule,
ChipController aChipController)

throws java.lang.IllegalArgumentException

描述：本方法用于设置根据所给 CAModule 进行解扰操作所缺省使用的芯片控制器。如果本方法没有被调用，在 CASModuleManager 的选择无需指定。

输入：

aModule-指定 CASModule

aChipController CASModule-所使用的缺省芯片控制器

输出：

无

异常处理：

java.lang.IllegalArgumentException 如果给定的 CASModule 尚未注册。

C.3.1.2.1.8 setCCIBits

语法：public void setCCIBits(CASModule aModule,
CASSession casSession,int cciBits)

描述：本方法用于设定终端对于本业务进行拷贝保护所需拷贝控制信息数据（CCI bits），CCI 比特信息的定义由终端软件平台指定并解释执行。

输入：

aModule-指定 CASModule

casSession-解扰请求会话

cciBits-当前服务使用的 CCI 比特值

输出：

无

C.3.1.2.1.9 setServiceListFilter

语法：public void setServiceListFilter(int filterData)

描述：本方法用于向终端软件平台提供用于业务列表过滤的参数，业务列表参数的具体含义由终端软件平台指定并执行。

输入：

filterData 业务列表过滤参数

输出：

无

C.3.1.2.1.10 registerCASPacketListener

语法：public void registerCASPacketListener(int casId,
CASPacketListener casPacketListener)
throws java.lang.IllegalArgumentException

描述：本方法用于 DCAS 应用注册一个 CAPacketListener，CAPacketListener 由终端软件平台调用，并向 DCAS 应用传送 CAS 数据包（如 EMM）。CA 系统标识由参数 casID 表示， CAS 数据包的接收由终端软件平台各自实现。

输入：

casId-CAsystemID

casPacketListener 需要注册的 CASPacketListener

输出：

无

异常处理：

java.lang.IllegalArgumentException 如果给定 casId 已经注册过 listener。

C.3.1.2.1.11 unregisterCASPacketListener

语法：public void unregisterCASPacketListener(CASPacketListener casPacketListener) throws java.lang.IllegalArgumentException

描述：本方法用于 DCAS 应用取消 CASPacketListener 的注册。

输入：

casPacketListener 需要取消注册的 listener

输出：

无

异常处理：

java.lang.IllegalArgumentException 如果给定 CASPacketListener 尚未注册。

C.3.1.2.1.12 DetachableSecurityDevice

语法：public DetachableSecurityDevice[] getDetachableSecurityDevices()

描述：本方法用于 DCAS 应用获得可分离设备的组件。

输入：

无

输出：

一个 DetachableSecurityDevice 对象

C.3.1.2.1.13 receiveOsdMsg

语法：public void receiveOsdMsg(byte[ ] msg, int[ ] flags)

描述：显示 OSD 信息，其参数的具体含义和具体项目相关。

输入：

msg – OSD 信息内容，可包括文本外的描述信息

flags – OSD 类型指示

输出：

无

C.3.1.2.1.14 receiveTuningAlert

语法：public void receiveTuningAlert (int[ ] serviceIdentifiers, int[ ] flags)

描述：应急广播。在某些项目中应急广播的参数不是通过 CA 系统来下发的，在此类情况下不必实现此函数。

输入：

serviceIdentifiers – 一组用于标示应急广播频道参数的数值。数值的含义由具体项目定义

flags – 可用于表示应急广播类型的参数

输出：

无

C.3.1.2.1.15 getCATNotifier

语法：public CATNotifier getCATNotifier()

描述：本方法被 DCAS 应用调用来获取 CAT notifer 对象，DCAS 应用可以通过 CAT notifier 注册获取 CAT 更新通知的监听器。DCAS 应用需要 CAT 信息来过滤带内 EMM。

输入：

无

输出：

CAT Notifier 对象

C.3.1.3 CA Descriptor接口

描述：本接口提供了 CA 描述符的信息，给定业务的 PMT 中的可能提供 CA 描述符。此外 CAT 中也会有 CA 描述符的出现。

C.3.1.3.1 方法

C.3.1.3.1.1 getCASystemId

语法：public int getCASystemId()

描述：本方法返回 CA 描述符的 CASystemId。

输入：

无

输出：

CASystemId

C.3.1.3.1.2 getCASystemId

语法：public int getPid()

描述：本方法返回 CA 描述符中的 PID (ECM PID 或 EMM PID)。

输入：

无

输出：

无

C.3.1.3.1.3 getPrivateData

语法：public byte[] getPrivateData()

描述：本方法返回 CA 描述符终端的私有数据数组。

输入：

无

输出：

privateData

C.3.1.4 CAServiceComponentInfo接口

描述：本接口用于提取特定 CA 业务成分的信息，如 ECM PID 和用于装载控制字的 DescramblerContext。

C.3.1.4.1 方法

C.3.1.4.1.1 getDescramblerContext

语法：public DescramblerContext getDescramblerContext()

描述：本方法返回用于DCAS应用装载控制字的DescramberContext对象，在PMT的组件循环中出现多次相同CA描述符（相同的ECMPID和私有数据）的情形下，应该只存在唯一一个DescramblerContext。

输入：

无

输出：

DescramblerContext

C.3.1.4.1.2 getCADescriptor

语法：public CADescriptor getCADescriptor()

描述：本方法返回业务组件相关的CA描述符，CADescriptor实例由PMT中的CA信息产生。

输入：

无

输出：

一个CADescriptor

C.3.1.4.1.3 getComponentStreamPIDs

语法：public int[] getComponentStreamPIDs()

描述：本方法返回一个由PMT中描述的基本流数组，数组元素的顺序应同getComponentStreamType返回的数组元素顺序一致。

输入：

无

输出：

ES（基本流）PID数组

C.3.1.4.1.4 getComponentStreamTypes

语法：public int[] getComponentStreamTypes()

描述：本方法返回PMT中的流类型数组，流类型值应遵循MPEG标准ISO13818-1。数组元素的顺序应同getComponentStreamPID返回的数组元素顺序一致。

输入：

无

输出：

流类型数组

C.3.1.4.1.5 getServiceIdentifiers

语法：public int[] getServiceIdentifiers()

描述：本方法返回对象所关联的业务标识数组，业务标识的表现形式由具体的使用环境而定。

输入：

无

输出：

ServiceID数组

C.3.1.5 CASPacketListener接口

DCAS应用通过本接口来接收带外CAS Packets（如EMMs）。

DCAS 应用根据给定的 CA 系统标识通过 CASModuleManager 类提供的 registerCasPacketListener 方法注册本监听器。

CA 系统标识由参数 casId 表示。

CAS 包的接收依赖终端软件平台来实现。

C.3.1.5.1 方法

C.3.1.5.1.1 casPacketArrived

语法：public void casPacketArrived(
int casId, byte [] casPacketData, byte [] casPacketHeader)

描述：DCAS 应用通过已注册的监听器来获得 CAS 包。

输入：

casId CA 系统标识

casPacketData CAS 包数据

casPacketHeader 依赖终端软件平台的 CAS 包头

输出：

无

C.3.1.6 CASPermission类

public class CASPermission extends java.security.BasicPermission

描述：

任一 DCAS 应用必须获取 CASPersmission 方可访问 CASModuleManager。

本机制用于确保只有被网络运营商授权的 DCAS 应用方可使用 DCAS API。

C.3.1.6.1 构造

C.3.1.6.1.1 CASPermission

语法：public CASPermission(String name)

描述：创建一个新 CASPermission. Name 字符串现不使用，应设为空字符串。

输入：

Name-本 CASPermission 的名称

输出：

无

C.3.1.6.1.2 CASPermission

语法：public CASPermission(String name, String actions)

描述：创建一个新 CASPermission. Name 字符串现不使用，应设为空字符串，actions 字符串现不使用，应设为空（null）。本构造方法用于 java.security.Policy 对象实例化一个新 Permission objects。

输入：

name 本 CASPermission 的名称

actions action 列表

输出：

无

C.3.1.7 CASSession接口

本接口提供CAS会话相关信息

```
public interface CASSession
public static final int TYPE_PRESENTATION = 0x00000001;
public static final int TYPE_RECORDING =    0x00000002;
public static final int TYPE_BUFFERING =    0x00000004;
```

C.3.1.7.1 方法

C.3.1.7.1.1 getType

语法：public int getType()

描述：本方法返回本会话的操作类型。

输入：

无

输出：

操作类型，可以是本接口中定义的值之一或组合

例如 – 本方法返回0x00000003，即是类型(0x00000001)和 (0x00000002)的组合

C.3.1.7.1.2 getNetworkInterface

语法：public org.davic.net.tuning.NetworkInterface getNetworkInterface()

描述：本方法返回同CAS会话相关联的NetworkInterface，DCAS应用可以从CAS会话中获取相关NetworkInterface对象。使用NetworkInterface, DCAS应用可得到先TransportStream对象，用于调用org.davic.mpeg.sections应用接口进行ECM Section过滤。

输入：

无

输出：

一个NetworkInterface对象

C.3.1.7.1.3 getAssociatedService

语法：public java.lang.Object getAssociatedService()

描述：本方法返回CAS会话相关的业务。

输入：

无

输出：

一个Service对象

C.3.1.7.1.4 getServiceContext

语法：public java.lang.Object getServiceContext()

描述：本方法返回CAS会话相关的ServiceContext。

注意：在某些中ServiceContext没有实际意义，本方法将返回空null。

输入：

无

输出：

一个ServiceContext对象

C.3.1.8 CAStatus接口

public interface CAStatus

DCAS 应用当调用 CASModuleManager 中的 sendDescramblingEvent 方法时使用本接口
DCAS 应用在每次 DescramblerContext 中的解扰状态发生变化时发送 CAStatus
本状态用来指示解扰成功与否
如果所解扰成分中任何一个发生解扰失败，本状态必须汇报整个业务的解扰请求失败
当终端软件平台收到一个新的 CAStatus,它应通过本段描述的 CAS 事件通知其他应用
具体应用接口在扩展应用接口部分说明

C.3.1.8.1 方法

C.3.1.8.1.1 isSuccess

语法：public boolean isSuccess()
描述：本方法返回解扰请求的状态。
输入：
无
输出：
如果解扰成功返回真。 false 任何原因解扰失败返回 false

C.3.1.8.1.2 getCAToken

语法：public int getCAToken()
描述：本方法返回用于其他应用通过 IXC 向 DCAS 应用查询网络相关信息的参数。
输入：
无
输出：
CA 令牌

C.3.1.9 CATListener接口

DCAS 应用需实现该接口，使用 CAT 中的 CA 描述符来过滤带内 EMM
DCAS 应用需要通过 CATNotifier 接口中定义的 registerCATListener 注册本监听器

C.3.1.9.1 方法

C.3.1.9.1.1 catUpdate

语法：public void catUpdate(CADescriptor desc, org.davic.net.tuning.NetworkInterface ni)
描述：本接口用于通知 DCAS 应用特定网络接口上的 CAT 更新。
DCAS 应用可以通过 NetworkInterface 对象，获取现 TransportStream 对象。
TransportStream 对象可用 org.davic.mpeg.sections 应用程序接口来实现 EMM section 过滤。
终端软件平台将 CAT 更新通知到 casId 相匹配，并且注册过的 CAT 监听器。
注意:
如果 CAT 不再被过滤（在成功过滤后）；或者 CA 描述符在 CAT 上被删除时，终端软件平台应当调用 catUpdate(null , theNetworkInterface)。
输入：

Desc-The CA descriptor. DCAS 应用通过 CASDescriptor 对象获取 EMM PID
ni CAT 更新所在的 NetworkInterface

输出：

无

C.3.1.10 CATNotifier接口

public interface CATNotifier
DCAS 应用使用本方法注册用于获取 CAT 更新通知的监听器
DCAS 应用使用 CAT 信息过滤带内 EMM

C.3.1.10.1 方法

C.3.1.10.1.1 registerCATListener

语法：public void registerCATListener(int casId, CATListener catListener)
描述：DCAS 应用调用本方法注册一个 CATListener。
输入：

casId CA 系统标识
catListener 由于注册的 CATListener

输出：

无

C.3.1.10.1.2 unregisterCATListener

语法：public void unregisterCATListener(CATListener catListener)
描述：DCAS 应用调用本方法取消注册某个 CATListener。
输入：

catListener 需要取消注册的 CATListener

输出：

无

C.3.2 终端软件平台底层API包

C.3.2.1 Key类

语法：public class Key
描述：表现一个基本密码密钥。用来决定 K-LAD 使用的密码算法及密码函数的输出参数。

C.3.2.1.1 构造

C.3.2.1.1.1 Key

语法：public Key(byte[] value, boolean encrypted)
输入：

value 密钥的值
encrypted-密钥是否被加密的标志，true 表示密钥已被加密，false 标识密钥是明文

C.3.2.1.2 方法

C.3.2.1.2.1 getKeyValue

语法：public byte[] getKeyValue()
描述：本方法返回密钥的值。
输入：
无
输出：
密钥的值

C.3.2.1.2.2 isEncrypted

语法：public boolean isEncrypted()
描述：本方法返回 true 时，标识密钥是加密的，false 表示密钥未加密。
输入：
无
输出：
true 密钥是加密的，false 表示密钥未加密

C.3.2.2 CWKey类

表现解扰密钥或控制字
public class CWKey extends Key;
public static final int PARITY_EVEN = 0;
public static final int PARITY_ODD = 1;

C.3.2.2.1 方法

C.3.2.2.1.1 CWKey

语法：public CWKey(byte[] value, boolean encrypted, int parity)
描述：
value 密钥的值。
真值标识密钥是加密的，假值表示密钥未加密。
奇偶值，表明控制字的奇偶性。

C.3.2.2.1.2 getParity

语法：public int getParity()
描述：本方法返回控制字的奇偶性。
输入：
无
输出：
控制字的奇偶性

C.3.2.3 DescramblerContext接口

public interface DescramblerContext
表现用来控制终端安全芯片解扰功能的组件
可以实例化多个 DescramblerContext 来使用不同密钥解扰多个码流的情形

C.3.2.3.1 方法

C.3.2.3.1.1 loadCW

语法：public void loadCW(int Vendor_SysID, CWKey cwKey, Key[] levelKeys, int schemeId) throws CADriverException

描述：本方法用于通知终端软件平台向解扰器装入控制字，并向终端安全芯片装入所需密钥。

一个解扰器通道是一个被单个控制字解扰的所有流的逻辑集合。

解扰器通道的使用依赖于 DescrablerContext。

此外，DCAS 应用应该通知终端软件平台当前控制字已失效（例如，由于授权原因），终端软件平台应当停止相应的解扰行为。

在这种情况下，DCAS 应用会提供一个 null CWKey。

输入：

Vendor_SysID：该值用于标识 CA 提供商在控制器中用于支持根密钥派生。终端安全芯片的根密钥由该值派生. 否则 Vendor_SysID 被忽略

cwKey：控制字. 如果控制字是明文，levelKeys 参数被忽略。如果 cwKey 为 null, 即 DCAS 应用没有提供有效的控制字

levelKeys：用于置入终端安全芯片的多级密钥。密钥数组的索引等于终端安全芯片中的绝对位置。在数组中特定元素值为 Null 表明终端安全芯片中相应位置不应装入密钥

即：levelKey[0]是 Key 1（被 Key 2 加密）；levelKey[1]是 Key 2（被 Key 3 加密）；levelKey[2]不必使用

schemeId：本 schemeId 用于指定终端安全芯片的加密算法(例如，AES，TDES)，ChipController 接口中定义了方式（scheme）值的列表。如果控制器只支持一种方式，则该值被忽略

输出：

无

异常处理：CADriverException 如果装入失败。

C.3.2.3.1.2 overrideChipController

语法：public void overrideChipController(ChipController aChipController) throws CADriverException

描述：本方法用于 DCAS 应用请求终端软件平台覆盖缺省终端安全芯片层级密钥的实现。可通过调用 CASModuleManager 的 setCurrentController 方法设置）。如果本方法没有被调用，终端安全芯片将使用缺省控制器。本方法只用于实现了多个终端安全芯片的终端安全芯片系统中。

输入：

aChipController 所要覆盖的控制器

输出：

无

异常处理:

CADriverException 如果操作失败。

C.3.2.4 chipcontroller芯片控制器接口

表现控制终端安全芯片执行的组件

public interface ChipController

C.3.2.4.1 字段

C.3.2.4.1.1 SCHEME_TDES

public static final int SCHEME_TDES=0
描述：用于指示终端安全芯片应使用 TDES 的值。

C.3.2.4.1.2 SCHEME_AES

public static final int SCHEME_AES=1
用于指示终端安全芯片应使用 AES 的值。

C.3.2.4.1.3 PROCESSING_MODE_REGULAR

public static final int PROCESSING_MODE_REGULAR=0
用于指示终端安全芯片认证应答算法中不需进行额外处理的值。

C.3.2.4.1.4 PROCESSING_MODE_POST_PROCESSING

Public static final int PROCESSING_MODE_POST_PROCESSING=1
用于指示终端安全芯片认证应答算法中需要实施后处理阶段的值。

C.3.2.4.2 方法

C.3.2.4.2.1 getPublicId

语法：public byte[] getPublicId() throws CADriverException
描述：本方法返回终端安全芯片的公共标识。
输入：
无
输出：
终端安全芯片的公共标识 publicId
异常处理：
CADriverException 如果访问终端安全芯片驱动过程中有通信错误。

C.3.2.4.2.2 getChipType

语法：public byte[] getChipType() throws CADriverException
描述：本方法返回终端安全芯片的类型标识。
输入：
无
输出：
终端安全芯片类型
异常处理：
CADriverException 如果访问终端安全芯片驱动过程中有通信错误。

C.3.2.4.2.3 getChipControllerProperty

语法：public java.lang.String getChipControllerProperty(java.lang.String propertyName) throws CADriverException
描述：本方法根据提供的终端安全芯片属性名称，返回该属性对应的值。本功能在本接口中被预留，可用于读出控制器中将来扩增的属性。现阶段没有定义任何属性名称。
输入：
propertyName 属性名称

输出：

属性值

异常处理：

CADriverException 如果访问终端安全芯片驱动过程中有通信错误。

C.3.2.4.2.4 authenticate

语法：public byte[] authenticate(int Vendor_SysID, byte[] challenge, Key[] levelKeys, int schemeId, int processingMode) throws CADriverExceptio

描述：本方法用于认证终端安全芯片中的层级密钥机制，终端安全芯片应根据送入的随机握手信息计算认证信息。

输入：

Vendor_SysID 该值用于标识 CA 提供商. 在控制器中用于支持根密钥派生。终端安全芯片的根密钥由该值派生. 否则 Vendor_SysID 被忽略

Challenge：握手信息，随机数

levelKeys：层级密钥所需各级密钥.密钥数组的索引等于终端安全芯片中的绝对位置。在数组中特定元素值为 Null 表明终端安全芯片中相应位置不应装入密钥。即：levelKey[0]是 null；levelKey[1]是 Key 2（被 Key 3 加密）；levelKey[2]不必使用

schemeId 本 schemeId 用于指定终端安全芯片的加密算法(例如，AES，TDES)

如果控制器只支持一种方式，则该值被忽略

processingMode：用于指定计算应答结果计算过程中是否实施额外

后处理过程的值，如果控制器只支持没有后处理模式，则该参数被忽略

输出：

终端安全芯片所计算的应答响应

异常处理：

CADriverException 如果访问终端安全芯片驱动过程中有通信错误。

C.3.2.4.2.5 encryptData

语法：

```
public void encryptData(int Vendor_SysID,
                        CWKey cwKey,
                        Key[] levelKeys,
                        int schemeId,
                        int encryptionId,
                        byte[] src,
                        int srcPos,
                        byte[] dest,
                        int destPos,
                    int length) throws CADriverException
```

描述：本方法调用芯片功能来加密内存中的数据。

输入：

Vendor_SysID 本参数用于标示 CA 提供商。安全芯片用此数值来派生根密钥

cwKey 加密用的控制字。如果控制字没有加密，之后的 levelKeys 将被忽略

levelKeys 层级密钥

数组中密钥的索引值等于其在层级密钥中的绝对位置。数组中的 Null 元素表明在此层级位置没有 key 需要被设置

schemeId 层级密钥所使用的加密算法。如果芯片只支持一种算法，则此参数将被忽略

encryptionId 数据加密/解密的算法（如 AES，TDES）。如果芯片只支持一种算法，则此参数将被忽略

src 源数据数组

srcPos 源数据数组的起始位置

dest 目的数据数组

destPos 目的数据数组的起始位置

length 需要处理的数据字节数

异常处理：

层级密钥通讯错误时，抛出 CADriverException 异常。

C.3.2.4.2.6 decryptData

```
语法：public void decryptData(int Vendor_SysID,
                              CWKey cwKey,
                              Key[] levelKeys,
                              int schemeId,
                              int encryptionId,
                              byte[] src,
                              int srcPos,
                              byte[] dest,
                              int destPos,
                              int length) throws CADriverException
```

描述：本方法调用芯片功能来解密内存中的数据。

输入：

Vendor_SysID 本参数用于标示 CA 提供商。安全芯片用此数值来派生根密钥

cwKey 解密用的控制字。如果控制字没有加密，之后的 levelKeys 将被忽略

层级密钥

数组中密钥的索引值等于其在层级密钥中的绝对位置。数组中的 Null 元素表明在此层级位置没有 key 需要被设置

schemeId 层级密钥所使用的加密算法。如果芯片只支持一种算法，则此参数将被忽略

encryptionId 数据加密/解密的算法（如 AES，TDES）。如果芯片只支持一种算法，则此参数将被忽略

src 源数据数组

srcPos 源数据数组的起始位置

dest 目的数据数组

destPos 目的数据数组的起始位置

length 需要处理的数据字节数

输出：

无

异常处理：

层级密钥通讯错误时，抛出 CADriverException 异常。

C.3.3 应用扩展API包

C.3.3.1 CASEventListener接口

本接口应被需要接收CAS事件的应用实现。

CAS events提供了当前ServiceContext的CA Status和基本信息

public interface CASEventListener

C.3.3.1.1 方法

C.3.3.1.1.1 receiveCASEvent

语法：public void receiveCASEvent(
CASEventInfo casEventInfo,
CASAppInfo casAppId,
boolean isSuccess,
int caToken)

描述：本方法用于向注册了CAS事件监听器的应用传递CAS事件。

输入：

casEventInfo CAS事件所属的信息

casAppId 用于标识发送事件的DCAS应用。这些标识可被应用通过IXC和DCAS应用通信。在没有DCAS应用可以解扰给定码流时，终端软件平台应当使用值null作为casAppId的值调用本方法，获取此种CAS事件通知的应用应根据自己的设计和实现来处理该情况

isSuccess 用于指示解扰是否成功的布尔值

caToken通过IXC传回DCAS应用的令牌，应用可使用该令牌通过IXC向DCAS应用查询特定的网络信息

C.3.3.2 类CASEventManager

public class CASEventManager

应用使用CASEventManager注册监听器，来获取CAS事件

CAS events提供了当前的CA Status和基本信息

C.3.3.2.1 方法

C.3.3.2.1.1 getInstance

语法：public static CASEventManager getInstance()

描述：本方法用于取得一个CASEventManager实例单体。

输入：

无

输出：

CASEventManager实例

C.3.3.2.1.2 addListener

语法：public void addListener(CASEventListener aCASEventListener)

描述：本方法用于应用注册一个CASEventListener。该监听器用于传递所有的的CAS事件。

输入：

aCASEventListener 需要注册的CASEventListener

输出：

无

C.3.3.2.1.3 removeListener

语法：public void removeListener(CASEventListener aCASEventListener)
描述：本方法用于应用取消注册一个 CASEventListener。
输入：
aCASEventListener 已经注册的 CASEventListener
输出：
无

C.3.3.3 CASAppInfo接口

本接口提供 DCAS 应用的信息
public interface CASAppInfo

C.3.3.3.1 方法

C.3.3.3.1.1 getAID

语法：public int getAID()
描述：本方法返回 DCAS 应用的 application ID。
输入：
无
输出：
DCAS 应用的 application ID

C.3.3.3.1.2 getOID

语法：public int getOID()
描述：本方法返回 DCAS 应用的 organization ID。
输入：
无
输出：
DCAS 应用的 organization ID

C.3.3.4 CASEventInfo接口

本接口提供 CASEvent 的信息

```
public interface CASEventInfo {
public static final int TYPE_PRESENTATION = 0x00000001;
public static final int TYPE_RECORDING =    0x00000002;
public static final int TYPE_BUFFERING =    0x00000004;
```

C.3.3.4.1 方法

C.3.3.4.1.1 getType

语法：public int getType()
描述：本方法返回产生 CAS Event 的操作类型。
输入：
无

输出：

操作类型，可以是本接口中定义的值其中之一或组合

例如 – 本方法返回 0x00000003，即是类型(0x00000001)和(0x00000002)的组合

C.3.3.4.1.2 getNetworkInterface

语法：public org.davic.net.tuning.NetworkInterface getNetworkInterface()

描述：本方法返回和 CAS Event 相关的 NetworkInterface。

输入：

无

输出：

一个 NetworkInterface 对象

C.3.3.4.1.3 getAssociatedService

语法：public java.lang.Object getAssociatedService()

描述：本方法返回 CAS Event 相关联的业务。

输入：

无

输出：

一个 Service 对象

C.3.3.4.1.4 getServiceContext

语法：public java.lang.Object getServiceContext()

描述：本方法返回同 CAS event 相关联的 ServiceContext。

注意，在某些操作中 ServiceContext 无实际意义。本方法返回 null。

输入：

无

输出：

一个 ServiceContext 对象

附　录　D
（资料性附录）
DCAS 用户端软件下载和启动的机制

DCAS 用户端软件下载和启动的安全机制是通过使用遵循 ITUT X.509 标准的数字证书链机制实现的。证书链中的根证书存储于终端软件平台中，通过各级证书的验证保证 DCAS 用户端软件的真实性和完整性。

D.1　证书生成

本标准不限制 DCAS 用户端软件数字证书签名层级数量，下面以运营商、条件接收系统和终端软件平台三层为例描述 DCAS 用户端软件数字证书的生成过程，如图 D.1 所示。

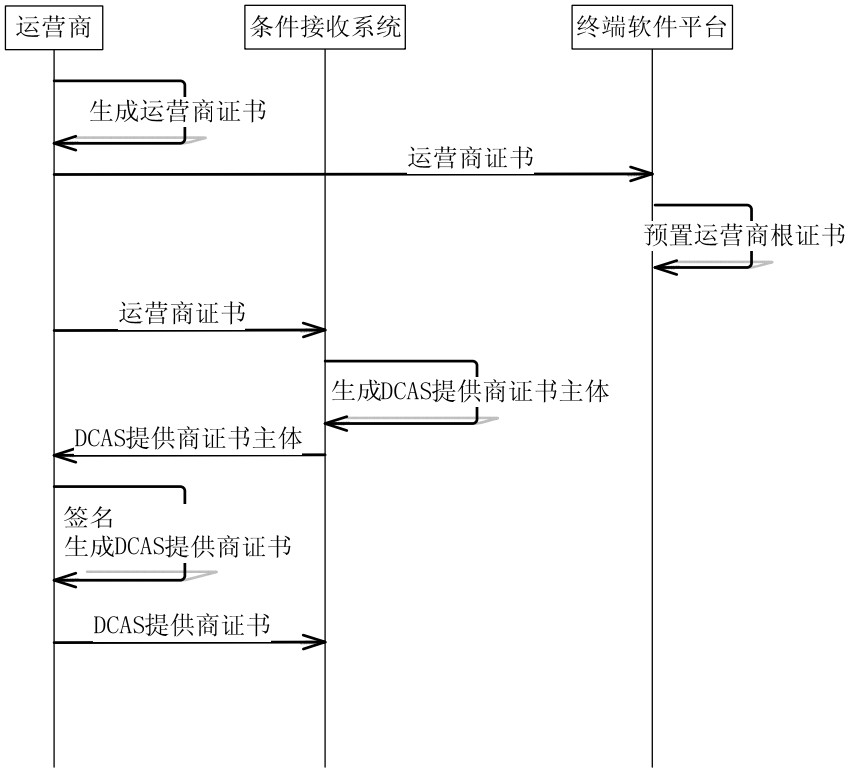


图 D.1　DCAS 用户端软件数字证书生成流程图

运营商生成运营商证书，并发送给终端软件平台厂商，终端软件平台厂商将此根证书预置到终端软件平台中。运营商安全地保存自己的私钥。

运营商向条件接收系统提供运营商证书。条件接收系统使用自己的公钥生成 DCAS 提供商证书主体，并发送给运营商进行签名，生成证书。

D.2　DCAS用户端软件的签名

以三层机制为例描述 DCAS 用户端软件的签名过程，如图 D.2 所示。

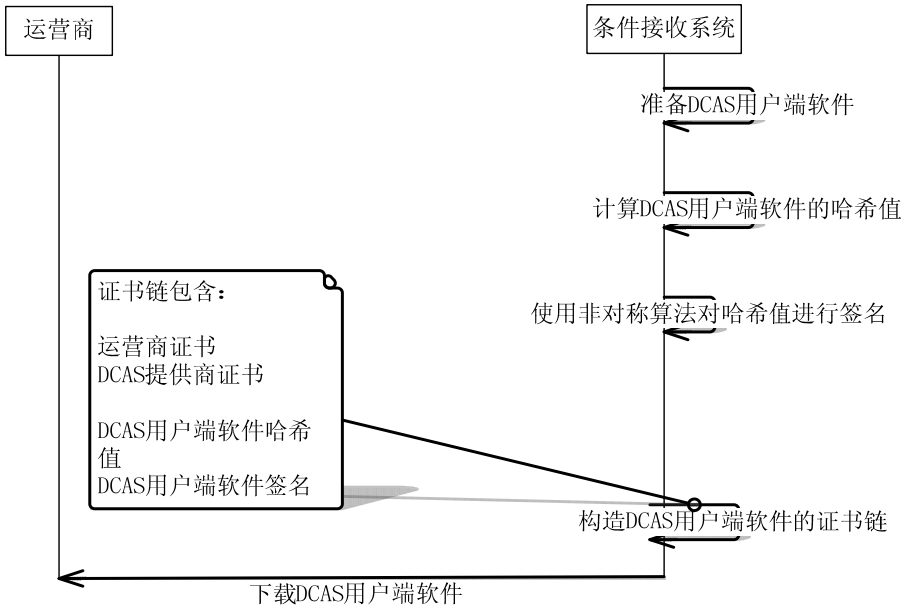


图 D.2　DCAS 用户端软件签名流程图

条件接收系统使用 Hash 算法计算需下发的 DCAS 用户端软件的哈希值。

条件接收系统使用自己的私钥，采用非对称算法对生成的 DCAS 用户端软件哈希值进行签名。

条件接收系统将包含有证书链、DCAS 用户端软件签名、DCAS 用户端软件哈希值和 Hash 算法的验证信息附加在 DCAS 用户端软件上，提供给运营商。其中：证书链包括运营商证书和 CAS 提供商证书，DCAS 用户端软件哈希值为可选项。

D.3　DCAS用户端软件验证

终端软件平台对 DCAS 用户端软件进行如下两种操作时，需进行验证：

当 DCAS 用户端软件下载到终端 RAM 后，存入终端 Flash 前对其进行签名验证。验证成功后，DCAS 用户端软件才可被存入终端 Flash 中。

当 DCAS 用户端软件装载至终端 RAM 后，对其进行签名验证。仅当验证成功后，DCAS 用户端软件才可被启动。

DCAS 用户端软件验证流程如图 D.3 所示。

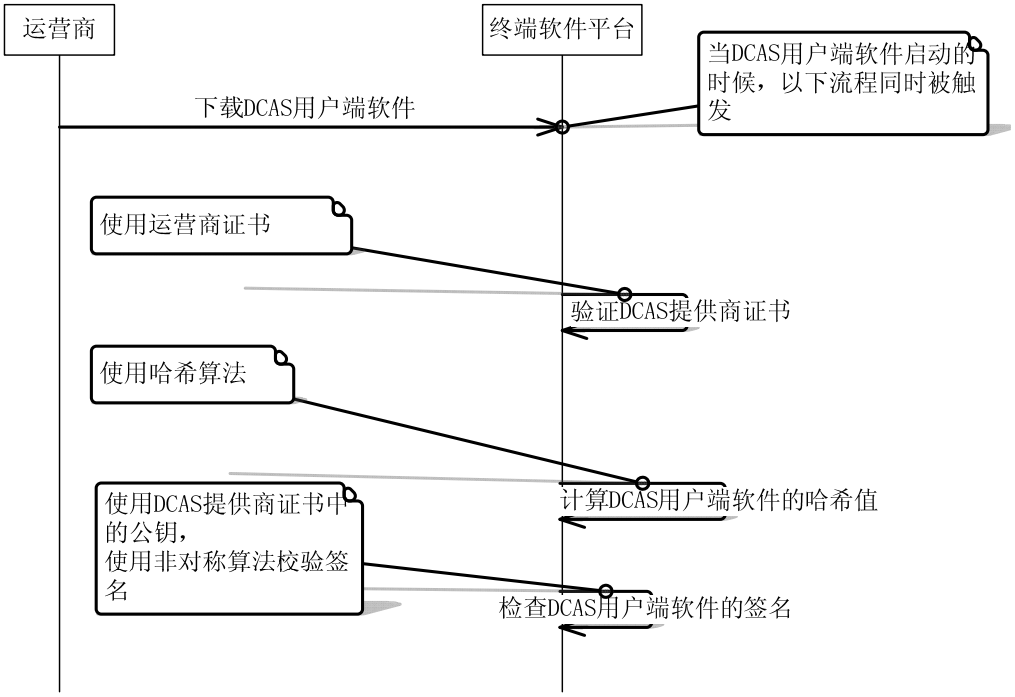


图 D.3　DCAS 用户端软件验证流程图

终端软件平台使用图中所述方法验证 DCAS 用户端软件的合法性：

a)　使用运营商证书中的公钥验证 CAS 提供商证书的签名；

b)　首先使用双方约定的摘要算法计算 DCAS 用户端软件的哈希值，再使用 CAS 提供商公钥以及非对称算法检查 DCAS 用户端软件签名信息。

若上述验证过程中的任一验证失败，则将该 DCAS 用户端软件丢弃。

D.4　DCAS用户端软件权限机制

为了控制 DCAS 用户端软件访问终端资源，前端应向终端发送 DCAS 用户端软件的访问权限文件。终端软件平台依据这些权限管理 DCAS 用户端软件执行。

DCAS 用户端软件访问权限文件由运营商签名，并附加上包含根证书和运营商证书的证书链。

终端软件平台只能接收由运营商签名的访问权限文件。终端软件平台应使用运营商证书检查签名。如果通过检查，则终端软件平台应解析 DCAS 用户端软件访问权限文件，并更新 DCAS 用户端软件相应权限。

附　录　E
（资料性附录）
DCAS 终端生产发放使用和软件签名认证流程

E.1　DCAS终端的生产发放使用流程

如图 E.1 所示，DCAS 终端的生产发放使用流程需要经过 6 个环节。

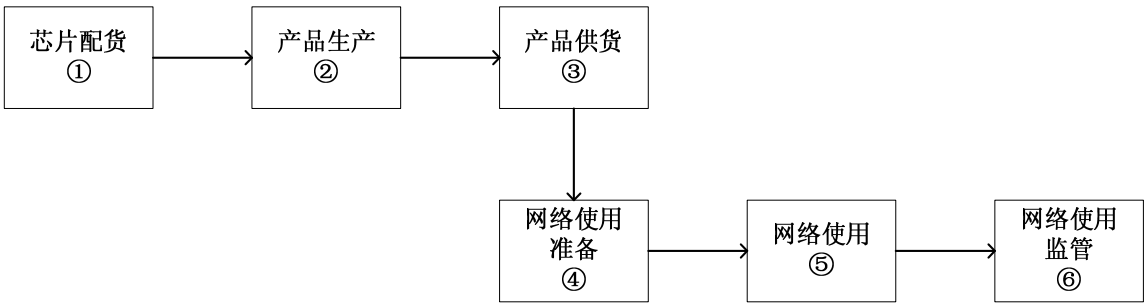


图 E.1　DCAS 终端生产发放使用流程

E.1.1　终端安全芯片备货

终端安全芯片备货参与方：安全数据管理平台、芯片厂商、终端厂商、运营商。

终端安全芯片备货流程：

a)　芯片厂商向 DCAS 安全数据管理平台提供芯片型号、数量、SCK 还原函数、SCK 初步处理函数信息；

b)　DCAS 安全数据管理平台向芯片厂商提供每个芯片相应的 ESCK 和 ChipID 等信息，并为每批次芯片提供 BL_Key0 信息；

c)　芯片厂商采用安全芯片密钥植入模块将 ESCK 和 ChipID 等以及 BL_Key0 信息植入安全芯片，完成安全芯片生产；

d)　终端厂商向芯片厂商下发芯片采购订单；

e)　芯片厂商向终端厂商提供芯片供货；

f)　终端厂商向运营商通告芯片备货情况。

E.1.2　产品生产

产品生产参与方：DCAS 安全数据管理平台、终端厂商、运营商。

产品生产流程:

a)　DCAS 安全数据管理平台向终端厂商提供 CA-KEY 生成算法、签名信息；

b)　终端厂商生产 DCAS 终端；对 ChipID 采用 CA-KEY 生成算法生成 CA-KEY；将签名信息烧入 Flash 写保护区域;灌入经过签名认证的启动加载软件、终端软件平台、DCAS 用户端软件；

c)　终端厂商导出 DCAS 安全数据管理平台所需信息（可省略）。

E.1.3　产品供货

产品供货参与方：终端厂商、运营商。

产品供货流程：终端厂商向运营商提供 DCAS 终端产品。

E.1.4 网络使用准备

网络使用准备参与方：终端厂商、DCAS 安全数据管理平台、条件接收厂商、运营商。

网络使用准备流程：

a) 芯片厂商向条件接收提供商/运营商提供 Seedv 和最终根密钥派生函数；

b) DCAS 安全数据管理平台向条件接收提供商/运营商提供 DCAS 前端所需的 Vendor_SysID、每个芯片的 SCKv；

c) 条件接收提供商/运营商将上述信息导入运营商的 DCAS 前端。

E.1.5 网络使用

网络使用参与方：运营商。

网络使用流程：DCAS 终端接入运营商网络后使用。

E.1.6 网络使用监管

网络使用监管参与方：终端厂商、运营商、条件接收提供商。

网络使用监管流程：

a) 终端厂商向运营商定期提供 CA-KEY 号使用情况月报；

b) 条件接收提供商监控运营商的 CAS 系统安全使用情况，为运营商提供安全保障；

c) DCAS 安全数据管理平台监管 DCAS 行业使用情况。

E.2 DCAS终端的软件签名认证流程

现有终端的软件签名认证流程需要经过以下几个环节，见图 E.2。

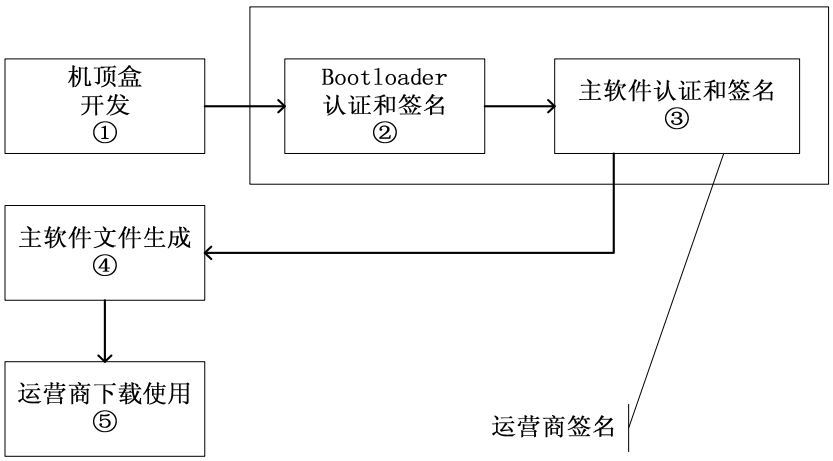


图 E.2 现有终端软件签名认证流程

DCAS 终端的软件签名认证流程将经过如下几个环节，见图 E.3。

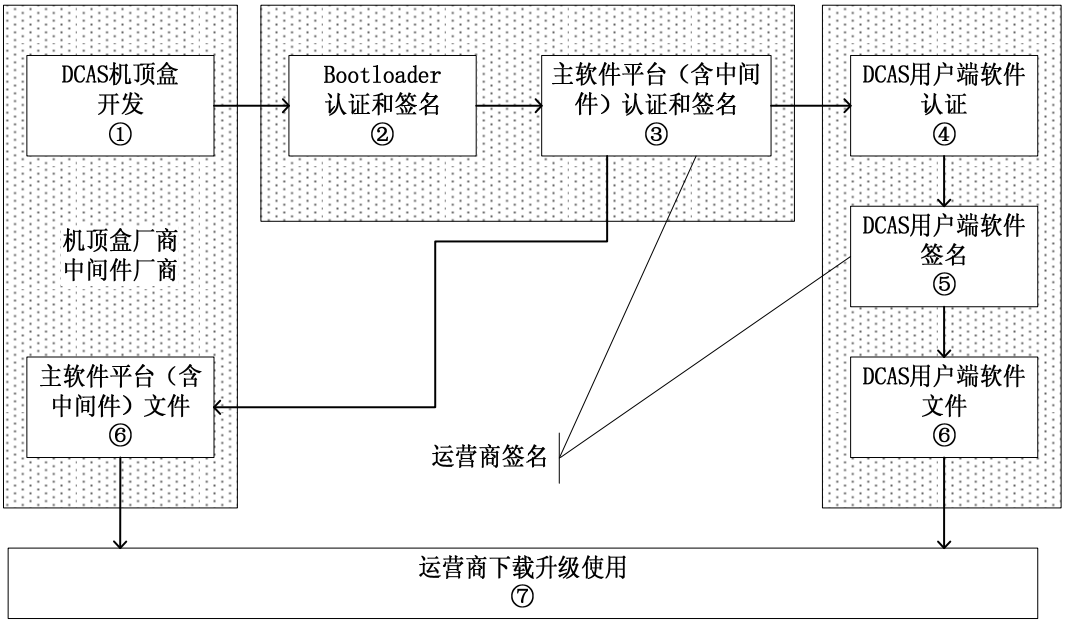


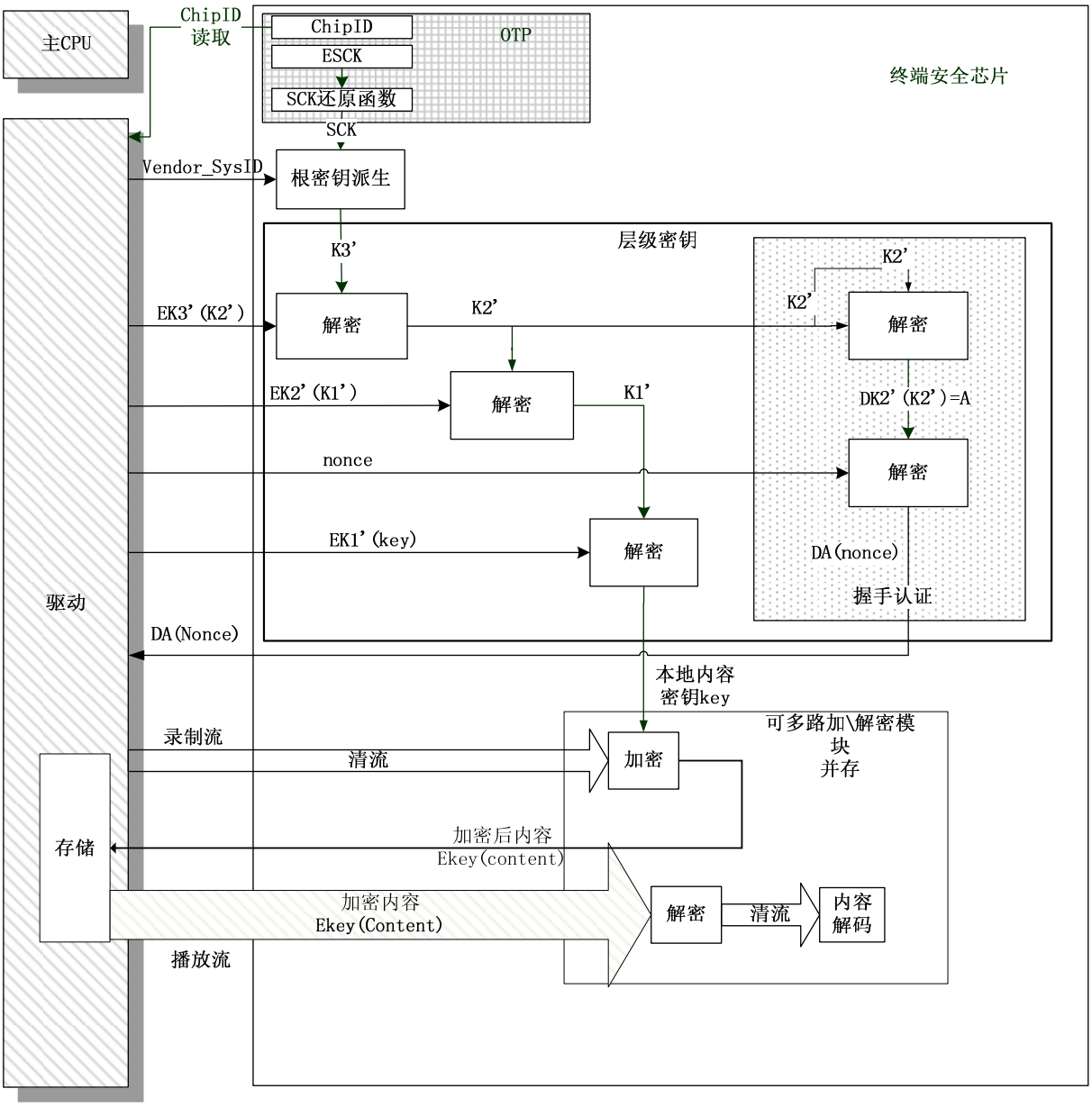
图 E.3　DCAS 终端的软件签名认证流程

附　录　F
（规范性附录）
终端安全芯片功能扩展

F.1　终端安全芯片对本地内容的保护

终端安全芯片应支持，在根密钥派生模块、层级密钥模块的基础上，提供用于对本地内容进行加扰的本地内容密钥以实现对本地内容保护的扩展功能。

本地内容保护流程见图F.1。



图F.1　本地内容保护流程

解扰后的清流应用本地内容密钥加密，加密后的本地内容可存储在终端的硬盘或者

Flash中。

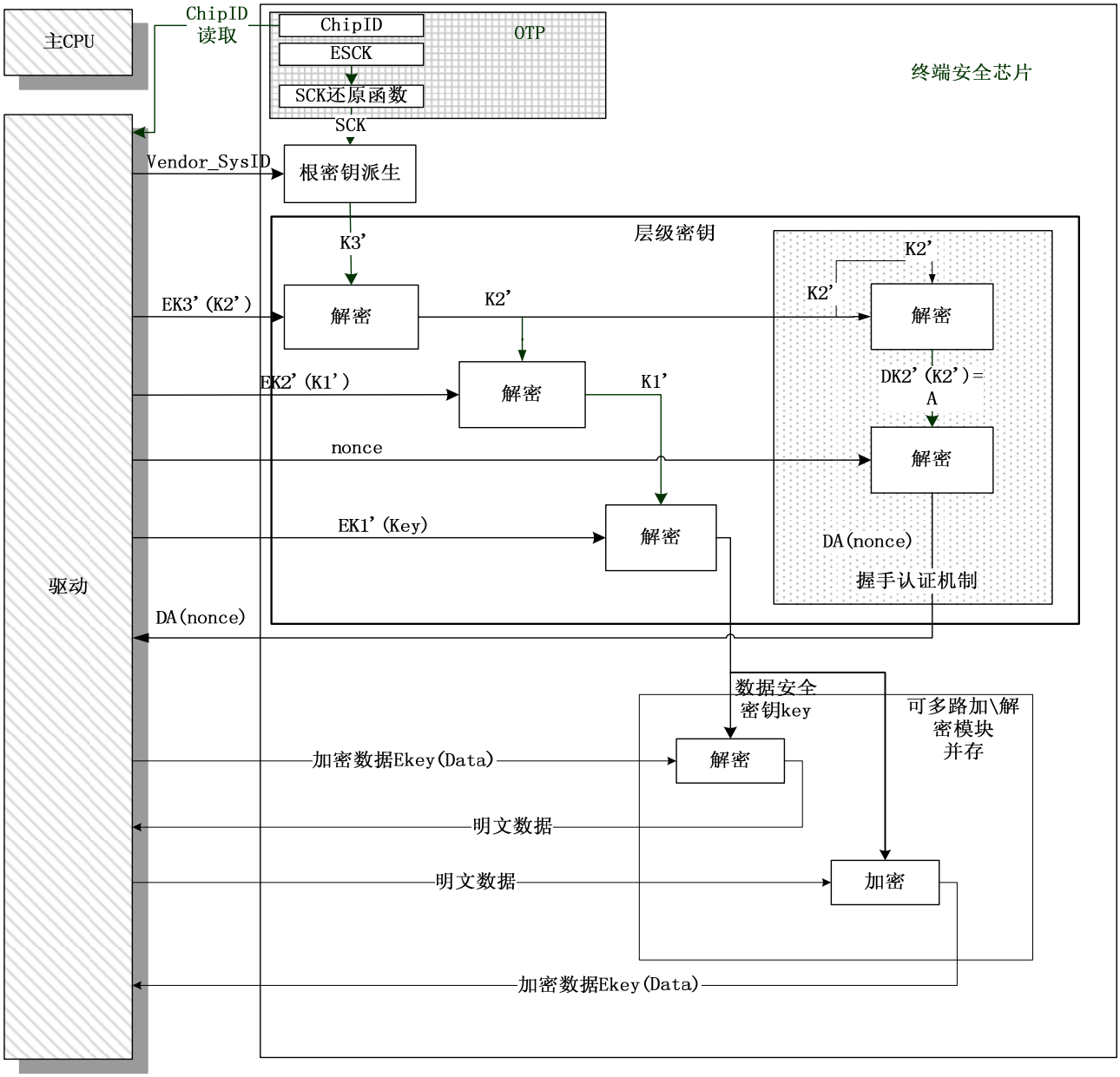
采用SCK可通过根密钥派生模块生成K3’，K3’的生成见F.3用于扩展功能的根密钥派生模块。

层级密钥按照9.3的要求根据K3’生成本地内容密钥。

F.2 终端安全芯片对数据安全的保护

终端安全芯片应支持，在根密钥派生模块、层级密钥模块的基础上，提供用于对数据进行加扰的数据安全密钥以实现对数据安全保护的扩展功能。

数据安全保护流程见图F.2。



图F.2 数据安全保护流程

明文数据通过数据安全密钥加密，加密后的数据存储在终端的硬盘或者Flash中。

数据安全密钥的生成方式同F.1本地内容密钥生成方式。

F.3 用于扩展功能的根密钥派生模块

见图F.3。

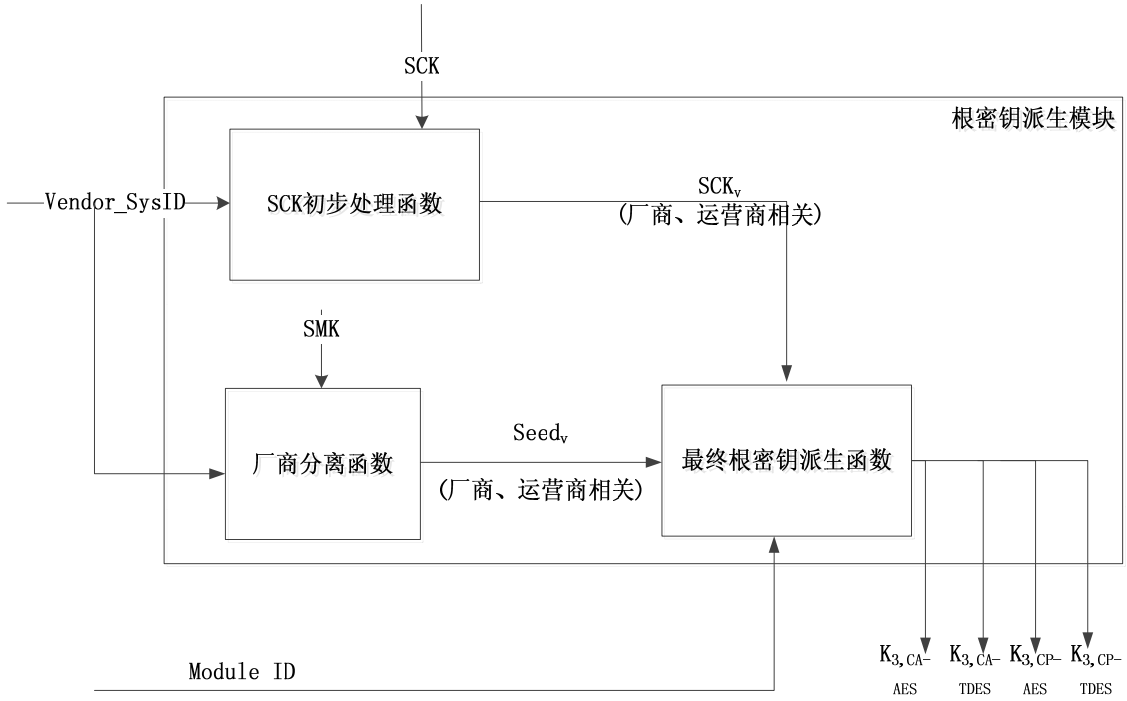


图 F.3 用于扩展功能的根密钥派生模块

用于扩展功能的根密钥派生模块可与终端安全芯片根密钥派生模块共用，根密钥派生模块的最终根密钥派生函数增加参数Module ID的输入，生成K3’。

Module ID是芯片驱动层的配置参数，典型的Module ID可使用以下数值之一：

a) CA-AES；

b) CA-TDES；

c) CP-AES (CP = Content Protection，业务保护)；

d) CP-TDES。

因此最终可生成4种根密钥：

a) K3,CA-AES；

b) K3,CA-TDES；

c) K3,CP-AES；

d) K3,CP-TDES。

在使用时：

K3, CA-AES必须作为3级AES层级密钥的根密钥，必须支持握手认证。从此层级密钥生成的控制字只能用于广播码流的解扰。

K3, CA-TDES必须作为3级TDES层级密钥的根密钥，必须支持握手认证。从此层级密钥生成的控制字只能用于广播码流的解扰。

K3, CP-AES必须作为3级AES层级密钥的根密钥，可选支持握手认证。从此层级密钥生成的控制字只能用于扩展功能的加密和解密。

K3, CP-TDES必须作为3级TDES层级密钥的根密钥，可选支持握手认证。从此层级密钥生成的控制字只能用于扩展功能的加密和解密。

附　录　G
（资料性附录）
根密钥派生模块相关要求

根密钥派生模块包括三个功能函数：SCK 初步处理函数，厂商分离函数和根密钥最终派生函数。芯片厂商对这些函数的实现方式限于如下两种：

a） 全部使用逻辑电路硬件实现；

b） 使用与应用程序CPU独立的安全CPU，以及在安全CPU内运行的可升级固件的方式实现。

芯片厂商对这些函数的实现，应用程序 CPU 应当不能访问、干涉其逻辑、状态和结果。

芯片厂商在这些函数实现过程中采用的算法，应当具有密码级的单向性。

在符合以上安全要求的前提下，芯片厂商可以采用不同的技术或者方式来实现根密钥派生模块。

获取安全数据管理平台的安全芯片认证时，芯片厂商需向安全数据管理平台提供相关技术资料，以证明其技术或者方式符合以上安全要求。

附　录　H
（规范性附录）
握手认证流程

H.1　DCAS前端与终端握手认证流程

DCAS 前端与终端握手认证流程见图 H.1。

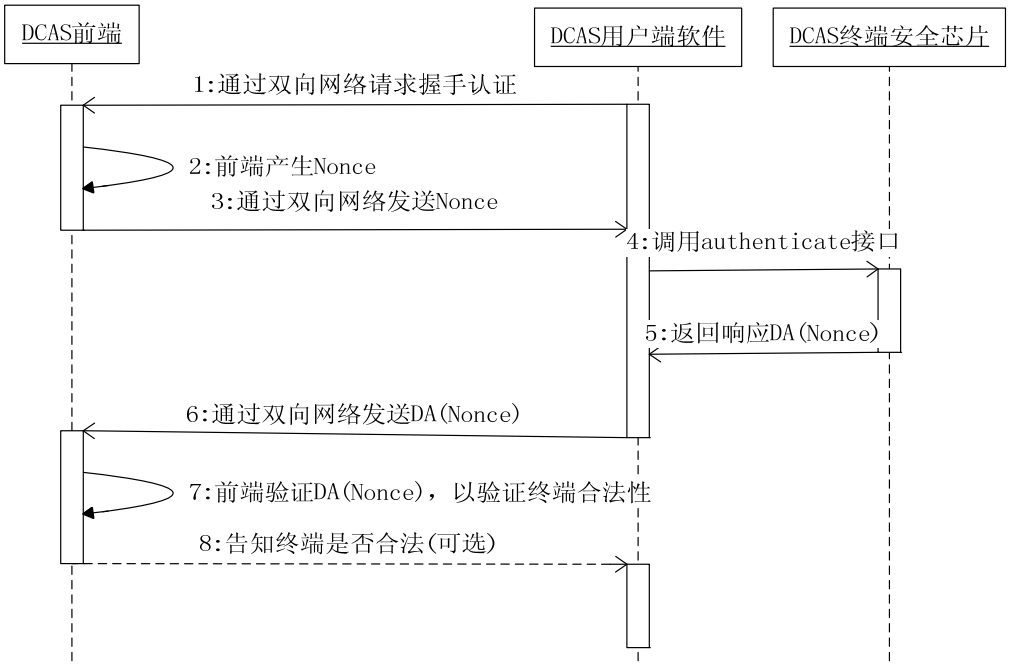


图 H.1　DCAS 前端与终端握手认证流程

H.2　DCAS前端产生Nonce并发送到终端安全芯片

a)　Nonce是16字节的随机数或伪随机数；
b)　Nonce通过双向网络传输。

H.3　DCAS终端安全芯片处理Nonce并将响应数据DA(Nonce)发送到前端

a)　终端安全芯片根据Vendor_SysID、EK3(K2)和Nonce生成响应DA（Nonce），处理参考9.3.2；
b)　DA(Nonce)也是通过双向网络传输。

H.4　前端收到终端响应数据DA(Nonce)后验证终端合法性

a)　前端使用与终端同样的算法，根据Nonce、K2也计算出一个DA(Nonce)，参考9.3.2。
b)　比对前端计算出来的DA（Nonce）和终端发来的响应DA（Nonce）值。如果相同，则验证通过，否则验证失败。

参 考 文 献

[1] ITUT X.509 Information technology－Open systems interconnection－The Directory: Public-key and attribute certificate frameworks

中　华　人　民　共　和　国
广播电影电视行业标准

可下载条件接收系统技术规范

GY/T 255－2012
*
国家广播电影电视总局广播电视规划院出版发行
责任编辑：王佳梅
查询网址：www.abp.gov.cn
北京复兴门外大街二号
联系电话：（010）86093424　　86092923
邮政编码：100866

版权专有　不得翻印