GY

中华人民共和国广播电视行业标准

GY/T 277—2019
代替 GY/T 277—2014

视音频内容分发数字版权管理技术规范

Technical specification of digital rights management for video audio content distribution

2019－07－05 发布　　2019－07－05 实施

国家广播电视总局　发布

目　　次

前　言

本标准按照GB/T 1.1—2009给出的规则起草。

本标准代替GY/T 277—2014《互联网电视数字版权管理技术规范》，与GY/T 277—2014相比，主要技术变化如下：

——修改了范围（见第1章，2014年版的第1章）；

——修改了规范性引用文件（见第2章，2014年版的第2章）；

——修改了术语和定义，将DRM代理修改为DRM客户端（见第3章，2014年版的第3章）；

——修改了概述（见5.1，2014年版的第5章）；

——修改了逻辑架构（见5.2，2014年版的第5章）；

——增加了内容授权（见5.3）；

——增加了密钥管理（见5.4）；

——修改了安全机制（见5.5，2014年版的9.2）；

——修改了信任模型（见5.6，2014年版的9.1）；

——增加了内容加密方法（见6.2）；

——修改了内容封装格式（见6.3，2014年版的第6章）；

——修改了许可证结构（见7.1，2014年版的7.1）；

——修改了许可证编码（见7.2，2014年版的7.2）；

——修改了许可证获取协议（见第8章，2014年版的第8章）；

——增加了DRM服务端（见第9章）；

——增加了DRM客户端（见第10章）；

——增加了数字证书格式、在线证书认证协议及证书撤销列表格式（见附录A）；

——修改了密码算法（见附录B，2014年版的附录A）；

——增加了DRM客户端功能接口（见附录C）；

——增加了DRM客户端运行环境接口（见附录D）；

——删除了基于HLS协议的流媒体中增加对ChinaDRM支持的说明（见2014年版的附录B）。

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：国家广播电视总局广播电视科学研究院、中央广播电视总台、深圳市海思半导体有限公司、英特尔（中国）有限公司、北京江南天安科技有限公司、北京数字太和科技有限责任公司、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、爱迪德技术（北京）有限公司、上海国茂数字技术有限公司、华数数字电视传媒集团有限公司、广东南方新媒体股份有限公司、中国传媒大学、北京安视网信息技术有限公司、百视通网络电视技术发展有限责任公司、湖南快乐阳光互动娱乐传媒有限公司、北京爱奇艺科技有限公司、阿里巴巴（中国）有限公司、辽宁广播电视台、上海文化广播影视集团有限公司、北京广播电视台。

本标准主要起草人：丁文华、郭沛宇、潘晓菲、王磊、田忠、梁志坚、吴迪、梅雪莲、赵云辉、赵鹏、饶玉、马吉伟、秦四春、薛子育、林卫国、赵志峰、郑黎方、张晶、田雪冰、刘好伟、刘琦、汪沛、范涛、陈靓、邵淇锋、冉大为、姜堑、汤毅、刘广宾、张智军、沈阳、张鹏、陈赫、陈钢、高宏鹏、吴南山、王立冬。

本标准于2014年5月首次发布。

引　言

本文件的发布机构提请注意，声明符合本文件时，可能使用涉及本文件有关内容的相关授权的和正在申请的专利如下：

序号	标准章条编号	专利名称	专利权利人
1	5.3	一种数字媒体内容保护方法及装置	广播电视科学研究院
2	5.3	一种数字媒体内容保护方法及装置、服务器、终端	广播电视科学研究院
3	7	一种数字媒体内容保护方法及装置	广播电视科学研究院
4	7	一种数字媒体内容保护方法及装置、服务器、终端	广播电视科学研究院

本文件的发布机构对于该专利的真实性、有效性和范围无任何立场。

该专利持有人已向本文件的发布机构保证，他愿意同任何申请人在合理且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文件的发布机构备案，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
广播电视科学研究院	北京市西城区复兴门外大街2号	孟祥昆	100866	010-86098010	mengxiangkun@abs.ac.cn

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

视音频内容分发数字版权管理技术规范

1 范围

本标准规定了视音频内容分发数字版权管理的逻辑架构、技术机制、内容加密、许可证格式、许可证获取协议、以及DRM服务端和DRM客户端的相关技术要求。

本标准适用于数字电视、IPTV、互联网电视等视音频内容分发过程中的版权保护。

2 规范性引用文件

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

GB/T 17964—2008 信息安全技术 分组密码算法的工作模式

GB/T 20518—2018 信息安全技术 公钥基础设施 数字证书格式

GB/T 32097—2016 信息安全技术 SM4分组密码算法

GB/T 32905—2016 信息安全技术 SM3密码杂凑算法

GB/T 32918.2—2016 信息安全技术 SM2椭圆曲线公钥密钥算法 第2部分：数字签名算法

GB/T 32918.4—2016 信息安全技术 SM2椭圆曲线公钥密钥算法 第4部分：公钥加密算法

GB/T 36322—2018 信息安全技术 密码设备应用接口规范

GY/T 257.1—2012 广播电视先进音视频编解码 第1部分：视频

GY/T 299.1—2016 高效音视频编码 第1部分：视频

ISO 14496-12:2015 信息技术 音视频对象编码 第12部分：ISO基础媒体文件格式（Information technology—Coding of audio-visual objects - Part 12: ISO base media file format）

ISO 23001-7:2016 信息技术 MPEG系统技术 第7部分：ISO基础媒体文件格式文件通用加密（Information technology—MPEG systems technologies - Part 7: Common encryption in ISO base media file format files）

ISO 23009-4:2013 信息技术 基于HTTP的动态自适应流媒体（DASH） 第4部分：分段加密与认证（Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 4:Segment encryption and authentication）

IETF RFC 2045 多目标INTERNET邮件扩展 第1部分：Internet消息主体格式（Multipurpose internet mail extensions—Part 1: Format of internet message bodies）

IETF RFC 2104 HMAC:用于消息验证的加密哈希（HMAC: Keyed-Hashing for Message Authentication）

IETF RFC 2560 X.509互联网公钥基础设施在线证书状态协议-OCSP（X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP）

IETF RFC 3279 互联网X.509公钥基础设施证书和证书撤销列表算法及标识（Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile）

IETF RFC 3280 互联网X.509公钥基础设施证书和证书撤销列表（Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile）

ECMA 404 JSON数据交换格式（The JSON data interchange format）

3 术语和定义

下列术语和定义适用于本文件。

3.1

内容提供者 content provider

拥有数字媒体内容并提供带版权信息的数字媒体内容的功能实体。

3.2

许可证 license

对数字媒体内容访问权限、使用规则和密钥等控制信息的描述。

3.3

设备 device

安装有DRM客户端的消费内容的实体。

3.4

DRM客户端 DRM client

设备中的可信实体，负责执行与DRM内容相关的许可和限制。

3.5

DRM服务端 DRM server

向DRM客户端提供许可证服务的实体。

3.6

DRM内容 DRM content

采用DRM技术管理的数字媒体内容。

3.7

密文 ciphertext

已加密的信息。

3.8

加密 encryption

为了产生密文，即隐藏数据的信息内容，由密码算法对数据进行（可逆）变换。

3.9

解密 decryption

与加密过程相对应的逆过程。即由密码算法对密文数据进行逆变换。

3.10

密钥 key

控制密码变换操作（例如：加密、解密、密码校验函数计算、签名生成或签名验证）的符号序列。

3.11

数字签名 digital signature

附加在数据单元上的一些数据，或是对数据单元所作的密码变换，用于验证数字信息来源的真实性和数据的完整性。

4 缩略语

下列缩略语适用于本文件。

CA 认证中心（Certification Authority）
CBC 密码分组链接（Cipher Block Chain）
CEI 内容加密信息（Content Encryption Information）
CEK 内容加密密钥（Content Encryption Key）
CENC 通用加密（Common Encryption）
ChinaDRM 中国数字版权管理（China Digital Rights Management）
CRL 证书撤销列表（Certification Revocation List）
DASH 用HTTP协议传输的动态自适应流媒体协议（Dynamic Adaptive Streaming over HTTP）
DER ASN.1的非典型编码规则（Distinguished Encoding Rules）
DRM 数字版权管理（Digital Rights Management）
HLS 基于HTTP的实时流媒体协议（Http Live Streaming）
HMAC 散列消息验证码（Hashed Message Authentication Code）
HTTP 超文本传输协议（Hyper Text Transport Protocol）
ISO 国际标准化组织（International Organization for Standardization）
IV 初始向量（Initialization Vector）
JSON JS对象简谱（JavaScript Object Notation）
MPD 媒体展现描述（Media Presentation Description）
NAL 网络抽象层（Network Abstract Layer）
OCSP 在线证书状态协议（Online Certificate Status Protocol）
PKI 公钥基础设施（Public Key Infrastructure）
PMT 节目映射表（Program Mapping Table）
SEI 辅助增强信息（Supplemental Enhancement Information）
TS 传送流（Transport Stream）
URI 通用资源标识符（Uniform Resource Identifier）
URL 统一资源定位符（Uniform Resource Locator）
UTC 协调通用时间（Coordinated Universal Time）
UUID 通用唯一识别码（Universally Unique Identifier）
uimsbf 无符号整数，高有效位优先（unsigned integer,most significant bit first）

5 体系架构

5.1 概述

本标准基于密码技术、PKI技术和授权技术定义了视音频内容分发数字版权管理端到端逻辑架构、内容授权、密钥管理、安全机制和信任模型。采用本标准定义的逻辑架构、技术机制、基础格式和协议，能够构造一个端到端的视音频内容分发数字版权管理系统。

5.2 逻辑架构

视音频内容数字分发数字版权管理系统从逻辑上分为DRM服务端和DRM客户端两个部分，如图1所示。DRM服务端系统包括内容加密、密钥管理、密钥网关和内容授权等核心模块。内容加密模块采用内容加密密钥（CEK）对视音频内容进行加密保护；密钥管理模块接收内容加密密钥后负责将该密钥同步给密钥网关；密钥网关模块在接收同步的密钥之后对内容加密密钥进行保密存储，并接收内容授权模块的密钥查询；内容授权模块接收DRM客户端的请求将包含有内容加密密钥和密钥使用规则的许可证安全的发送到合法的DRM客户端。DRM客户端接收到许可证后，按照密钥使用规则合理的解密内容加密密钥，并用内容加密密钥解密内容进行播放。

DRM服务端各模块、DRM客户端等基于PKI技术建立信任关系，基于此信任关系进行彼此之间的安全通信。

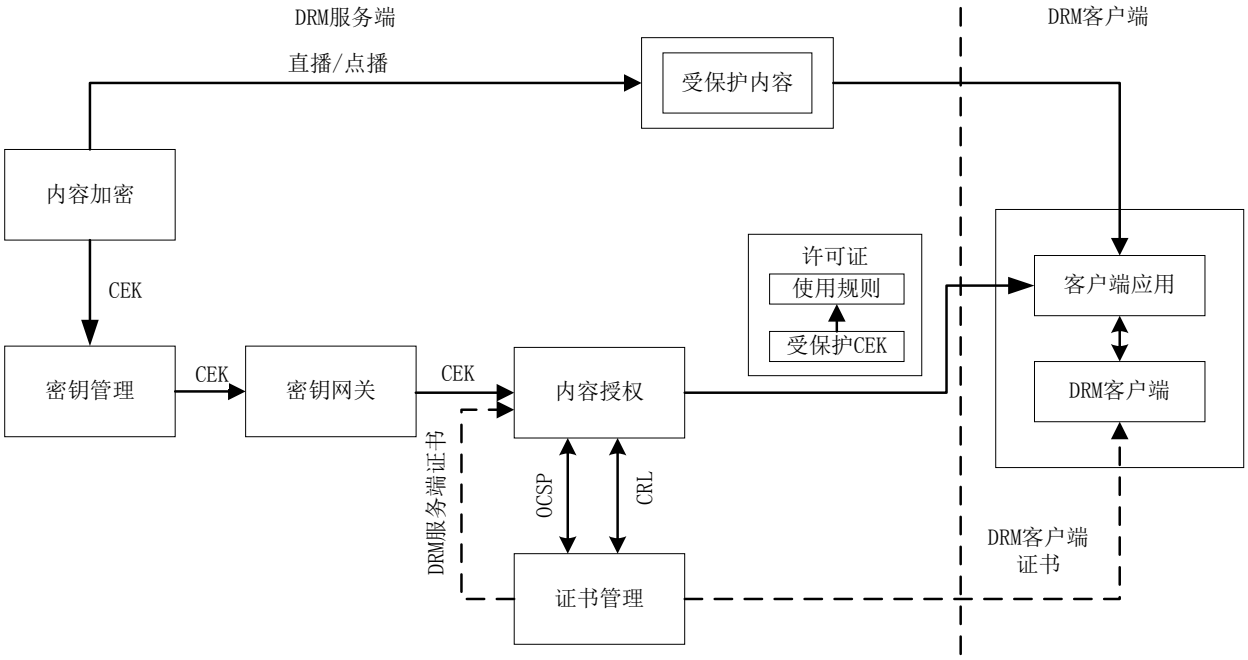

图1 视音频内容分发数字版权管理逻辑架构

5.3 内容授权

视音频内容分发数字版权管理系统基于层级密钥与密钥使用规则关联的机制实现内容的许可授权。逻辑上，密钥按照其加密保护的顺序可分为多个层级，加密当前密钥的密钥称为上级密钥。每一层级的密钥都有对应的密钥使用规则，当前密钥只能在上级密钥使用规则规定的条件下进行解密；每一个密钥又可能有多个密钥使用规则，密钥只能在满足其所有使用规则的前提下才能够被使用，密钥与密钥使用规则关联的机制如图2所示。

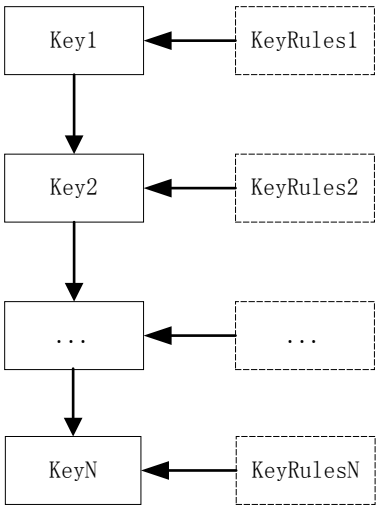

图2　密钥与密钥使用规则关联机制

密钥使用规则一般包括起始时间、截止时间、时间段、次数、累计时间段、输出规则、客户端安全等级等，不同类型的密钥也可能包括特殊的密钥使用规则。

起始时间表示该密钥在某规定时间之后才允许使用。

截止时间表示该密钥在某规定时间之前才允许使用。

时间段表示该密钥在第一次使用之后的某个时间段内允许使用。

次数表示使用该密钥的次数，使用该密钥成功完成一次解密计1次，如果该密钥是用来解密数字媒体内容的密钥，则默认为成功完成解密播放计1次，即DRM客户端完成1次内容解密并安全退出计1次。

累计时间段是对客户端播放器从播放到停止的时间的累加，如果多次播放到停止的时间超过累计时间段，则不再允许使用该密钥；累计时间段一般用于预览的场景，如一部影片时长90min，累计允许播放10min，则用户可以选择播放影片任意位置的内容，但累计不能超过10min。

输出规则规定使用内容加密密钥解密播放内容后，已解码内容数据是否允许输出到其他设备，以及允许的输出范围和方式，没有输出规则时则不限制输出方式。

客户端安全等级要求表示采用内容加密密钥解密播放内容时只允许在特定安全级别的DRM客户端进行解密播放，客户端安全等级要求分为软件安全级别、硬件安全级别、增强硬件安全级别，客户端安全等级要求保存在DRM客户端证书中，只有DRM客户端证书中的安全等级等于或者高于密钥使用规则中规定的客户端安全等级要求时，才可采用内容加密密钥进行内容的解密播放或输出。

如果没有对应的密钥使用规则，则表示该密钥的使用没有限制条件。

视音频内容分发数字版权管理系统中可能包含多种类型的密钥，如内容加密密钥、设备密钥、会话密钥、消息验证码密钥：

a)　内容加密密钥

内容加密密钥是加密数字媒体内容的密钥；一个内容可能有多个内容加密密钥。内容加密密钥的密钥使用规则包括：起始时间、截止时间、时间段、次数、累计时间段、输出规则、客户端安全等级要求等。

b)　会话密钥

会话密钥是客户端申请许可证时生成的临时密钥，该密钥用于加密保护内容加密密钥。

c)　消息验证码密钥

消息验证码密钥是生成用于保护许可证完整性的消息验证码的临时密钥。

d)　设备密钥

设备密钥指的是DRM客户端的密钥对，设备密钥应为非对称密钥，采用设备公钥加密的数据只有DRM客户端能够解密。

DRM服务端采用内容加密密钥加密数字媒体内容，将内容加密密钥和其它与内容相关的密钥（如会话密钥、消息验证码密钥等）采用层级密钥加密的方法加密后与各密钥对应的密钥使用规则一起打包成内容授权许可证发送给DRM客户端，DRM客户端按照层级密钥体系中各级密钥的使用规则使用密钥，实现对数字媒体内容的解密播放。视音频内容分发数字版权管理系统的内容授权机制如图3所示。

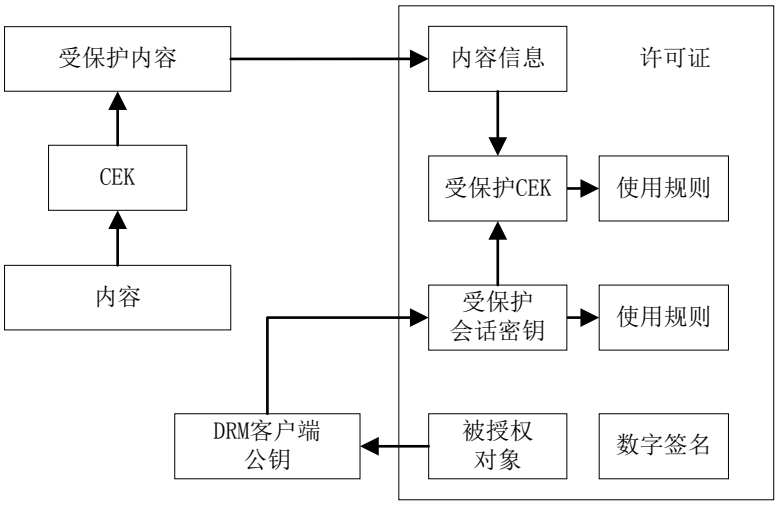

图3　内容授权机制

5.4　密钥管理

视音频内容分发数字版权管理系统DRM服务端和DRM客户端拥有各自的非对称密钥对，基于非对称密码算法进行许可证获取。DRM服务端采用对称密码算法进行内容加密，将内容加密密钥加密后封装在许可证中发送给DRM客户端。视音频内容分发数字版权管理系统密钥管理机制如图4所示。

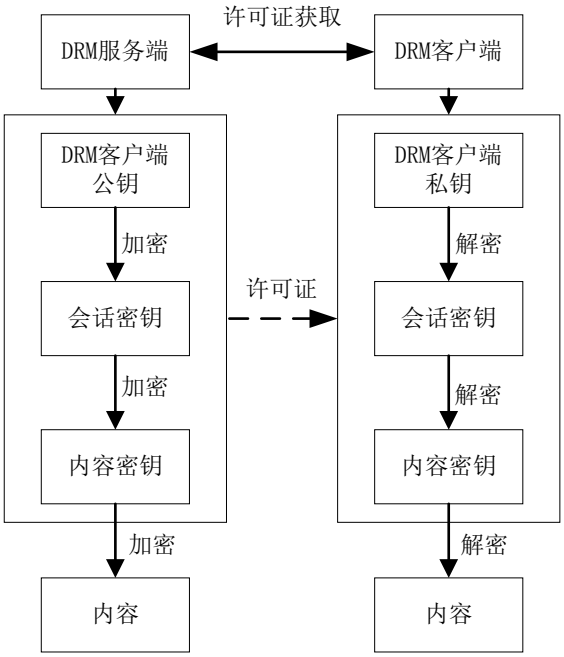

图4　密钥管理机制

DRM服务端和DRM客户端证书中包含其自身公钥,DRM服务端和DRM客户端分别安全保护自己的证书私钥。

视音频内容采用对称密码算法加密；DRM服务端生成会话密钥，用该会话密钥加密内容加密密钥；DRM服务端用DRM客户端公钥加密会话密钥；如果内容加密密钥需要同步到密钥网关，则采用密钥网关公钥加密；DRM服务端将加密后的会话密钥、加密后的内容加密密钥封装在许可证中发送给DRM客户端；许可证采用消息验证码保障许可证的完整性。

DRM客户端接收到许可证后，用DRM客户端私钥解密出会话密钥，然后用解密出的会话密钥解密内容加密密钥；DRM客户端解密得到内容加密密钥后，用内容加密密钥解密内容，实现内容的解码播放。

5.5　安全机制

本标准规定的视音频内容分发数字版权管理安全机制如下：

a)　数据机密性

数据机密性应通过加密保护敏感数据，敏感数据至少包括受保护内容和内容加密密钥。

b)　身份鉴别

身份鉴别应通过DRM客户端和DRM服务端之间验证对方的数字证书有效性来实现。

c)　数据完整性

数据完整性的检测应通过协议消息数字签名和许可证上的消息验证码进行验证。

5.6　信任模型

视音频内容数字版权管理系统的信任模型基于PKI体系。DRM系统中DRM服务端各核心组件、DRM客户端等都向认证中心申请获得一个数字证书，作为自己身份的凭证，互相之间的信任关系基于数字证书的有效性。如果DRM客户端的证书被DRM服务端验证有效，则DRM服务端信任该DRM客户端。

数字证书是DRM系统建立信任体系的基础。DRM客户端需要与数字证书进行关联。每个DRM客户端应至少携带一个数字证书。每个DRM客户端必须有唯一标识符，此唯一标识符应当在数字证书的适当字段载入。

视音频内容分发数字版权管理系统信任链包括根CA证书、DRM服务端子CA证书、DRM客户端子CA证书、DRM服务端证书、DRM客户端证书、以及OCSP服务器证书。安全信任机制如图5所示。

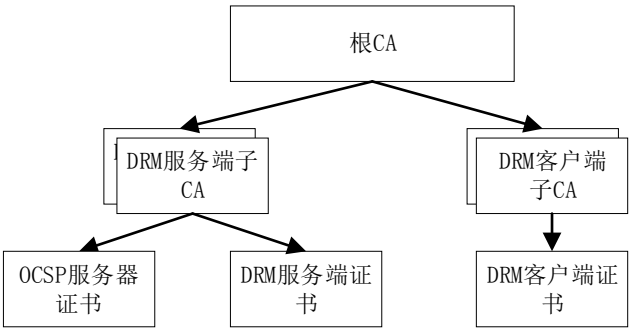

图5　安全信任机制

信任链建立以后，DRM服务端安全的存储DRM服务端证书及私钥、OCSP服务器证书、DRM服务端子CA证书、根CA证书；DRM客户端安全存储DRM客户端证书及私钥、DRM客户端子CA证书、根CA证书。

DRM服务端基于DRM客户端证书CRL列表判断DRM客户端证书的有效性，DRM客户端基于OCSP响应判断DRM服务端证书的有效性。

本标准规定视音频内容分发数字版权管理系统中的数字证书格式、在线证书认证协议以及证书撤销列表格式见附录A。

6　内容加密

6.1　概述

本章规定视音频内容分发数字版权管理系统的内容加密方法和加密内容封装格式。加密后的内容应包含内容标识及获取许可证所必需的信息。根据不同的应用场景，需要定义不同的加密内容封装格式。

6.2　内容加密方法

6.2.1　概述

视音频内容分发数字版权管理系统对视频内容基本码流进行加密，在基本码流的扩展数据中增加内容加密信息CEI用来指明随后的视频是如何被加密的。CEI中包括加密标识符、当前内容加密密钥标识符、下一个内容加密密钥标识符以及当前内容加密密钥对应的初始向量等。在下一个CEI出现之前，所有的视频数据按照6.2.2～6.2.4规定的方式进行加密，每个视频帧中需要加密的数据均采用当前CEI中的初始向量。CEI数据语法格式见表1。

表1 CEI 数据语法格式

语法	位数
CEI_DATA () {	
encryption_flag	1
next_key_id_flag	1
reserved	5
if (encryption_flag == 1) {	
current_key_id	128
}	
if (next_key_id_flag == 1) {	
next_key_id	128
}	
IV_length	8
IV	IV_length×8
}	

encryption_flag：标识该扩展数据之后的视频数据是否加密。

next_key_id_flag：标识该CEI数据中是否包含下一密钥标识符。

current_key_id：标识随后的视频加密用到密钥标识符。

next_key_id：标识视频加密将用到的下一个密钥标识符；这样在密钥变换之前，终端可以提前向服务器申请这个密钥。

IV_length：加密算法的初始向量长度。

IV：加密算法的初始向量。

6.2.2 广播电视先进音视频编解码与高效音视频编解码

在GY/T 257.1—2012和GY/T 299.1—2016规定的视频码流格式中，CEI数据放在sequence_header后的extension_data中。以下以GY/T 299.1—2016中的语法格式进行描述，CEI数据存放位置如下：

```
extension_data( i ) {
    while ( next_bits(32) == extension_start_code ) {
    extension_start_code
    if (i == 0) { /*  序列头之后  */
        if (next_bits(4) == '1101')      /*  CEI  */
        CEI()
      }
    }
}
```

CEI语法见表2。

表2 CEI 语法

语法	位数	说明
CEI ( ) {		
extension_id	4	‘1101’
reserved	4	
CEI_DATA()		见表 1
}		

GY/T 299.1—2016规定的视频编码内容的加密是在extension_and_user_data(0)中包含CEI数据的扩展信息，对每帧数据中的条带进行加密，加密后仅保留第一个条带的头信息，同一视频序列中的各帧起始加密使用同一初始向量，即使用该视频序列中CEI扩展信息中包含的初始向量。GY/T 299.1—2016规定的视频编码内容加密语法如下：

```
video_sequence{
    do{
        sequence_header()
        extension_and_user_data(0)
        do{
            if(next_bits(32) == intra_picture_start_code) {
                intra_picture_header()
            else
                inter_picture_header()
            extension_and_user_data(1)
            picture_data()//保留第一个 slice()头部，其余部分加密。
        }while(next_bits(32)==inter_picture_start_code||next_bits(32)==intra_picture_start_code)
    }while(next_bits(32) != video_sequence_end_code && next_bits(32) != video_edit_code)
    if(next_bits(32) == video_sequence_end_code)
            video_sequence_end_code
    if(next_bits(32) == video_edit_code)
            video_edit_code
}
```

符合广播电视先进音视频编解码、高效音视频编解码内容加密封装示意图如图6所示。

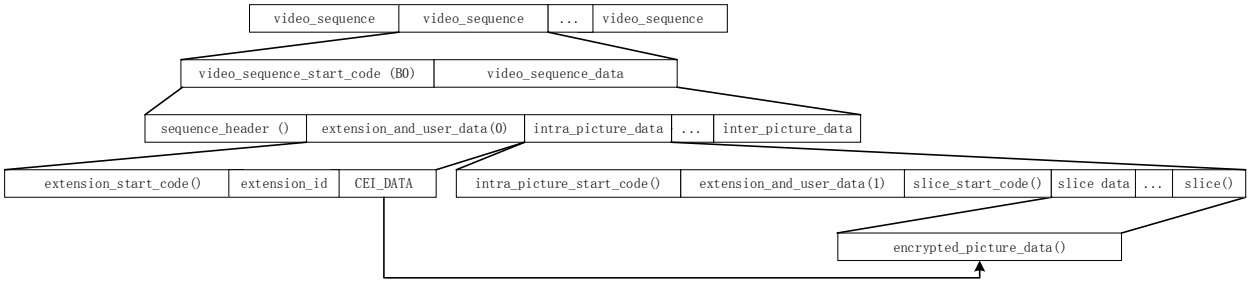

图6　内容加密封装示意图

内容加密分为全加密模式和部分加密模式两种。

全加密模式是指对条带的数据部分进行加密，最后小于等于16个字节的部分不加密，加密算法采用SM4算法，加密模式采用CBC模式,语法如下：

```
Encrypted_slice_data ()
{
   while (bytes_remaining() > 16)
   {
      protected_block // 16 bytes * n
   }
   unencrypted_trailer // 1-16 bytes
}
```

部分加密模式是对条带的数据进行10%加密，即加密一个16字节块之后，后面9个16字节块不加密，最后小于等于16字节的数据不加密，加密算法采用SM4算法，加密模式采用CBC模式，语法如下：

```
Encrypted_slice_data ()
{
   while (bytes_remaining() > 0)
   {
       if (bytes_remaining() > 16) {
           protected_block // 16 bytes
       }
       unencrypted_trailer  // // MIN(144, bytes_remaining()) bytes
   }
}
```

6.2.3　H.264

针对H.264编码的视频内容，本标准规定在NAL单元类型为6的SEI信息里面包含CEI扩展信息，对nal-unit-type为1或5的NAL单元进行加密，其他 NAL 单元不加密。在SEI中的User-data_unregistered用来包含CEI信息。user_data_payload_byte指明CEI信息；UUID是固定的值：70c1db9f-66ae-4127-bfc0-bb1981694b66。

H.264编码的视频内容加密分为全加密模式和部分加密模式两种。全加密模式是指对条带的数据部分进行加密，最后小于等于16字节的部分不加密，加密算法采用SM4算法，加密模式采用CBC模式，语法如下：

```
Encrypted_NAL_Unit ()
{
    NAL_unit_type_byte // 1 byte
    unencrypted_leader // 31 bytes
    while (bytes_remaining() > 16) {
        protected_block          // 16 bytes
    }
    unencrypted_trailer           // 1-16 bytes
}
```

部分加密模式是对条带的数据进行10%加密，即加密一个16字节块之后，后面9个16字节块不加密，最后小于等于16字节的数据不加密，加密算法采用SM4算法，加密模式采用CBC模式，语法如下：

```
Encrypted_NAL_Unit ()
{
    NAL_unit_type_byte // 1 byte
    unencrypted_leader // 31 bytes
    while (bytes_remaining() > 0) {
        if (bytes_remaining() > 16) {
            encrypted_block           // 16 bytes
        }
        unencrypted_block        // MIN(144, bytes_remaining()) bytes
    }
}
```

6.2.4 H.265

针对H.265编码格式视频内容，在NAL单元类型为39的SEI信息里面包含CEI扩展信息，对nal-unit-type 0-31的数据加密，其他 NAL 单元不加密。在SEI中的User-data_unregistered用来包含CEI信息。在user_data_payload_byte指明CEI信息；UUID是固定的值：70c1db9f-66ae-4127-bfc0-bb1981694b66。

H.265编码的视频内容加密分为全加密模式和部分加密模式两种。全加密模式是指对条带的数据部分进行加密，最后小于等于16个字节的部分不加密，加密算法采用SM4算法，加密模式采用CBC模式，语法如下：

```
Encrypted_NAL_Unit ()
{
    NAL_unit_type_byte // 1 byte
    unencrypted_leader // 64 bytes
    while (bytes_remaining() > 16) {
        protected_block           // 16 bytes
    }
    unencrypted_trailer           // 1-16 bytes
}
```

部分加密模式是对条带的数据进行 10%加密，即加密一个 16 字节块之后，后面 9 个 16 字节块不加密，最后小于等于 16 字节的数据不加密，加密算法采用 SM4 算法，加密模式采用 CBC 模式，语法如下：

```
Encrypted_NAL_Unit ()
{
    NAL_unit_type_byte // 1 byte
    unencrypted_leader // 64 bytes
    while (bytes_remaining() > 0) {
        if (bytes_remaining() > 16) {
            encrypted_block           // 16 bytes
        }
        unencrypted_block        // MIN(144, bytes_remaining()) bytes
    }
}
```

6.2.5 防止起始码的二义冲突

本条规定 CEI_DATA、加密后的编码数据为防止起始码二义冲突，需要进行转换，转换规则如下：

00 00 00 换成 00 00 03 00

00 00 01 换成 00 00 03 01

00 00 02 换成 00 00 03 02

00 00 03 换成 00 00 03 03

在解密之前需要将 CEI_DATA、有加密处理的单元等做相应还原再解密。即：

00 00 03 00 换成 00 00 00

00 00 03 01 换成 00 00 01

00 00 03 02 换成 00 00 02

00 00 03 03 换成 00 00 03

6.3 内容封装格式

6.3.1 TS 封装方式

TS 封装方式是指以数字电视或 IPTV 等传输流方式传输内容时的加密内容封装格式,TS 封装方式的内容以 HLS 协议分发时应遵循 6.3.4 的规定。为了终端播放器能够识别加密流，获取加密流中的 DRM

信息，需要在PMT表中增加DRM描述子。DRM描述子存在于PMT表中，主要包括加密内容格式、加密方式、许可证获取URI等信息，DRM描述子语法见表3。

表3 DRM描述子语法

语法	位数
ChinaDRM_descriptor（）{	
descriptor_tag	8
descriptor_length	8
video_format	4
video_encryption_method	4
audio_format	4
audio_encryption_method	4
DRM_data_bytes	（descriptor_length-2）×8
}	

descriptor_tag固定为0xC0。

video_format：表示该码流中加密内容的视频编码格式，见表4。

表4 视频编码格式

编码格式	规定
广播电视先进音视频编解码	0001
高效音视频编解码	0010
H.265	0011
H.264	0100
保留	0000,0101～1111

video_encryption_method：表示该码流中视频内容加密的方式，见表5。广播方式采用SM4-CBC加密方式。

表5 视频加密方式

加密方式	规定
NONE	0000
SM4-SAMPLE	0001
SM4-CBC	0011
保留	0100～1111

DRM_data_bytes：指明获取许可证的URI信息。由于descriptor_length是一个字节，这样最大的URI的尺寸是253。

6.3.2 MPEG DASH

基于本标准的数字版权管理系统在使用ISO 23009-4:2013 MPD文件描述时，在ContentProtection的@schemeIdUri属性定义为“3d5e6d35-9b9a-41e8-b843-dd3c6e72c42c”；在ContentProtection的

@Value属性中应按“ChinaDRM版本/ChinaDRM方案提供商/扩展信息”的方式赋值，通过“/”进行区分，如“ChinaDRM V2.0/CompanyName/Extended Information”。

6.3.3 CENC

通用加密格式CENC是基于ISO 14496-12:2015的一种加密格式，这种加密方式使不同的DRM系统能够解密同一个文件，CENC具体规定见ISO 23001-7:2016。

基于本标准的视音频内容分发数字版权管理系统在使用CENC通用加密格式时，应符合以下要求：

a） 在ProtectionSystemSpecificHeaderBox（‘PSSH’）中，将16字节长度的SystemID的设置为“3d5e6d35-9b9a-41e8-b843-dd3c6e72c42c”。

b） 在PSSH的Data部分，应包含获取许可证的URL。

c） 对TrackEncryptionBox（‘tenc’）中的default_IsEncypted或SampleGroupDescriptionBox（‘sgpd’）中的IsEncypted的设置增加对加密算法SM4-CBC的支持，具体定义如下：

——0x0:没有加密。

——0x1:加密。

- 采用SM4_CBC加密时，保护模式信息盒（‘sinf’）中的模式类型盒（‘schm’）中的模式类型scheme_type=’sm4c’；
- 采用SAMPLE-SM4加密时，保护模式信息盒（‘sinf’）中的模式类型盒（‘schm’）中的模式类型 scheme_type=’ sm4s’ 。

——0x000002～0xFFFFFF:保留。

6.3.4 HLS

HLS中加密媒体分块的加密密钥通过#EXT-X-KEY来指定，格式如下：

#EXT-X-KEY:<attribute-list>。

属性包括METHOD、VIDEOFORMAT 、URI、IV、KEYFORMAT，属性说明见表6。

表6 属性说明

属性	说明	是否强制
METHOD	用字符串来标识媒体加密方法：NONE、SAMPLE-SM4、SM4-CBC；NONE表示该内容未加密，这种情况下不允许出现 URI、IV、KEYFORMAT、KEYFORMATVERSIONS等属性	如果#EXT-X-KEY存在则该属性必须存在
VIDEOFORMAT	加密文件视频格式，支持:257.1、299.1、H.265、H.264	METHOD非NONE情况下必须存在
URI	获取许可证的URI字符串	METHOD非NONE情况下必须存在
IV	十六进制整数，代表加密初始向量	可选
KEYFORMAT	按“ChinaDRM版本/ChinaDRM方案提供商/扩展信息”的方式赋值，通过“/”进行区分，如“ChinaDRM V2.0/CompanyName/Extended Information”	METHOD非NONE情况下必须存在

7 许可证格式

7.1 许可证结构

数字版权管理的许可证由内容、被授权对象、密钥、密钥使用规则和数字签名等逻辑元素构成，如图7所示。

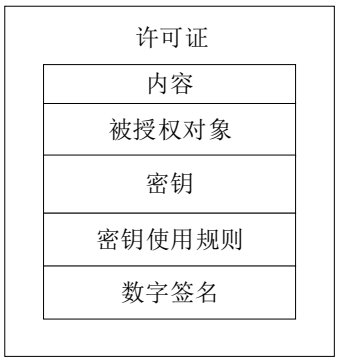

图7　许可证结构

许可证的元素说明如下：

a）　内容

内容是数字化的事物，例如图、文、音频、视频、动漫及其集合（如文件、数据库、软件、可执行代码等）等，一切可以数字化的事物都是内容。

b）　被授权对象

被授权对象是对指定内容相关权利的承载者。被授权对象通过其唯一标识描述。

c）　密钥

密钥是指该许可证中所包含的密钥信息，包括密钥的类型、算法、密钥数据等。

d）　密钥使用规则

密钥使用规则包括密钥索引信息以及起始时间、截止时间、时间段、次数等相关规则信息。在使用密钥时应按照密钥使用规则中包含的规则合理使用密钥。

e）　数字签名

数字签名单元是对其前面所有单元数据进行签名，保证数据的完整性。

7.2　许可证编码

7.2.1　编码方法

许可证编码由许可证索引单元和一系列的基本单元组成。许可证索引单元描述许可证的版本、许可证ID和基本单元的数量。基本单元包括：内容单元、被授权对象单元、密钥单元、密钥使用规则单元、以及数字签名单元等。许可证发布时由许可证索引单元和后续的一个或多个基本单元组成，许可证索引单元应为许可证的第一个单元，许可证编码结构如图8所示。

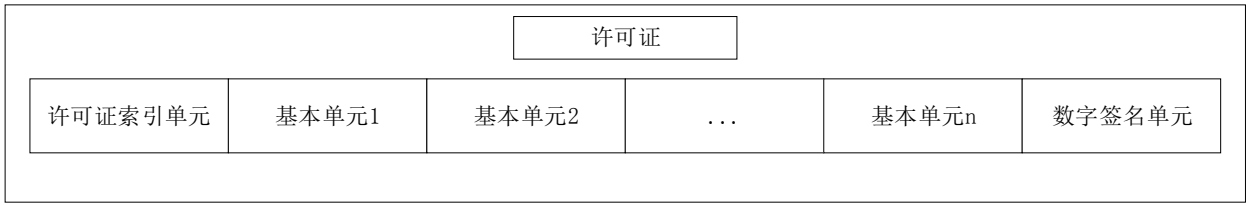

图8　许可证编码结构

许可证中的许可证索引单元和基本单元均由单元标识、长度和数据三个部分组成，单元编码方法如图9所示。

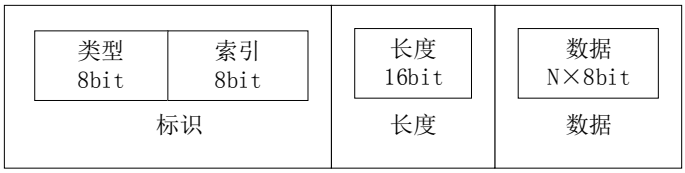

图9　单元编码方法

标识由2个字节构成，包括类型和索引，第1个字节是类型，第2个字节是该单元在许可证中的索引，用于支持许可证的分段传输。单元的索引从0开始，许可证索引单元为许可证的第1个单元，许可证索引单元编码后的索引始终为0；许可证索引单元后的基本单元的索引依次为1、2、3、……，依此类推。

长度是该单元实际数据信息的长度，由两个字节表示。

数据是单元的实际数据。

单元类型规定见表7。

表7　单元类型

类型	编码
许可证索引	0x00
内容	0x01
被授权对象	0x02
密钥	0x03
密钥使用规则	0x04
数字签名	0xFF
保留	0x05～0xEF

一个许可证由多个独立的不同单元构成，单元的数量和种类由许可证索引单元标识；被授权对象单元包含的是被授权方的信息。内容单元中包含密钥单元的唯一标识，内容单元中可对应多个密钥单元；密钥使用规则单元通过其中的密钥单元唯一标识与密钥单元对应。密钥单元通过上层密钥标识符与加密该密钥的密钥单元唯一对应。

本标准规定DRM服务端生成随机会话密钥，该密钥为对称密钥，用该密钥加密内容加密密钥CEK；采用客户端公钥加密会话密钥；生成随机消息验证码密钥，用该消息验证码密钥生成HMAC实现对许可证的完整性保护；采用客户端公钥加密消息验证码密钥。本标准规定的密钥单元包括内容加密密钥单元、会话密钥单元、消息验证码密钥单元等。数字签名单元是许可证的最后一个单元，一个许可证只有一个数字签名单元。数字签名单元采用消息验证码方式保护许可证的完整性和合法性，许可证框架如图10所示。

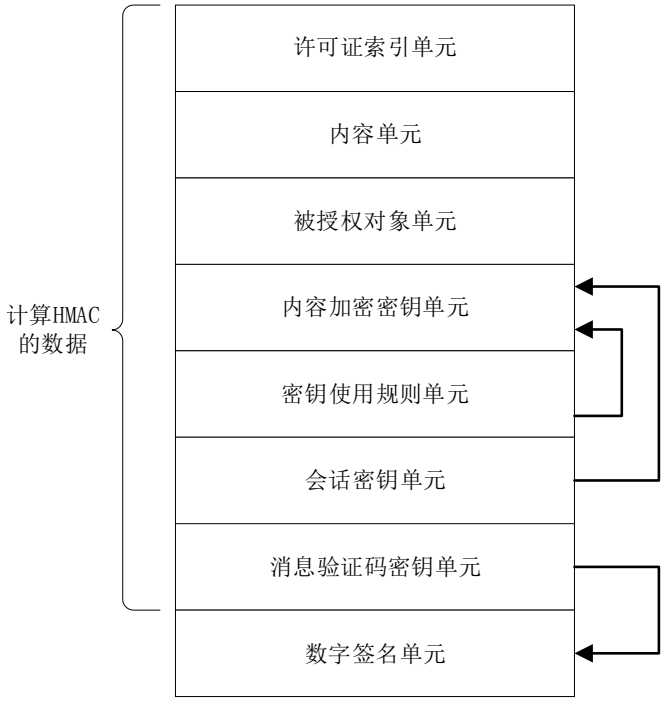

图10　许可证框架

7.2.2　许可证索引单元

许可证索引单元是许可证的第一个单元，许可证索引单元中包括：版本、许可证编号、基本单元数量、时间戳；许可证单元数据结构见表8。

表8　许可证索引单元数据结构

字段	比特数	类型	描述
Type	8	uimsbf	0x00
Index	8	uimsbf	0x00
Length	16	uimsbf	数据长度
Version	8	uimsbf	许可证版本
LicenseID	64	uimsbf	许可证 ID
UnitsNumber	8	uimsbf	许可证中基本单元的数量
TimeStamp	32	uimsbf	时间戳

Version：许可证版本号，当前为0x02。

LicenseID：许可证的唯一编号。

UnitsNumber：许可索引单元后面包含的基本单元的数量。

TimeStamp：许可证生成时的当前时间，该时间为UTC时间，用32位长度uimsbf类型数据表示，表示自1970年1月1日0:00:00起到该时间点的秒数。

7.2.3　内容单元

内容单元类型为0x01，具体数据结见表9。

表9　内容单元数据结构

字段	比特数	类型	描述
Type	8	uimsbf	0x01
Index	8	uimsbf	0x01～0xFF
Length	16	uimsbf	数据长度
ContentIDLen	8	uimsbf	内容 ID 长度
ContentID	N	uimsbf	内容唯一标识
CEKCount	8	uimsbf	该内容中包含的内容加密密钥个数
for(i< CEKCount){			
KeyIdentifierLen	8	uimsbf	密钥标识长度
KeyIdentifier[]	N	uimsbf	密钥标识
}			

Type：单元类型，内容单元类型为0x01。

ContentIDLen：内容唯一标识长度。

ContentID：内容唯一标识。

CEKCount:该内容的内容加密密钥个数。

KeyIdentifierLen、KeyIdentifier：密钥标识用来唯一标识密钥，密钥唯一标识为UUID；密钥标识长度用1个字节表示。KeyIdentifierLen、KeyIdentifier个数等于CEKCount。

7.2.4　被授权对象单元

被授权对象是对指定内容相关授权的承载者。被授权对象单元中包括被授权对象类型和被授权对象ID，被授权对象单元数据结构见表10。

表10　被授权对象单元数据结构

字段	比特数	类型	描述
Type	8	uimsbf	0x02
Index	8	uimsbf	0x01～0xFF
Length	16	uimsbf	数据长度
ObjectType	8	uimsbf	被授权对象类型
ObjectID	8×N	uimsbf	被授权对象唯一标识

ObjectType: 被授权对象的类型，默认为0。

ObjectID：被授权对象唯一标识，该唯一标识用设备密钥唯一标识来表示。

7.2.5　密钥单元

密钥单元中包括密钥算法、密钥数据、密钥类型、密钥标识、加密该密钥的密钥类型和密钥标识信息。密钥单元数据结构见表11。

表11　密钥单元数据结构

字段	比特数	类型	描述
Type	8	uimsbf	0x03
Index	8	uimsbf	0x01～0xFF
Length	16	uimsbf	数据长度
KeyAlgorithm	8	uimsbf	密钥算法
KeyDataLen	16	uimsbf	密钥数据的长度
KeyData[]	M	uimsbf	密钥数据
KeyType	8	uimsbf	密钥类型
KeyIdentifierLen	8	uimsbf	密钥标识长度
KeyIdentifier[]	N	uimsbf	密钥标识
UpperKeyType	8	uimsbf	上层密钥类型
UpperKeyIdentifierLen	8	uimsbf	上层密钥标识长度
UpperKeyIdentifier[]	N	uimsbf	上层密钥标识

KeyAlgorithm：密钥算法，用1个字节标识该密钥所对应的加密算法。这里指的应该是加密该密钥所使用的算法，而不是用该密钥加密别的内容时的算法。如：内容加密密钥单元中的密钥算法指的是加密CEK所采用的算法，而不是用CEK加密内容时采用的加密算法；因为采用CEK加密内容时采用的加密算法已在内容封装格式中具体进行了定义，并且在内容加密封装格式中传输。密钥算法规定见附录B。

KeyDataLen、KeyData[]：密钥数据是由加密该密钥的密钥加密后的密钥数据。

KeyType：密钥类型，用1个字节标识该密钥的功能，密钥类型规定见表12。

KeyIdentifierLen、KeyIdentifier：密钥标识用来唯一标识该密钥；本标准只定义密钥标识的长度和密钥标识数据；KeyIdentifierLen表示密钥标识的长度，用1个字节表示；KeyIdentifier表示密钥标识。

UpperKeyType：加密该密钥的密钥类型，密钥类型规定见表12。

UpperKeyIdentifierLen、UpperKeyIdentifier：加密该密钥的密钥的唯一标识。

表12　密钥类型

类型	编码	描述
内容加密密钥	0x01	加密内容用的密钥
设备密钥	0x03	客户端密钥，即 DRM 客户端的公钥，设备密钥单元可将 KeyData 设置为空，KeyDataLen 为 0
会话密钥	0x20	临时密钥的一种，许可证中用于加密保护密钥的临时密钥
消息验证码密钥	0x21	临时密钥的一种，许可证中用于计算消息验证码的临时密钥
保留	其他	保留

7.2.6　密钥使用规则单元

密钥使用规则规定被授权对象要按照密钥使用规则合理的使用密钥。密钥使用规则单元中包括：密钥标识、使用规则；密钥使用规则单元数据结构见表13。

表13　密钥使用规则单元数据结构

字段	比特数	类型	描述
Type	8	uimsbf	0x04
Index	8	uimsbf	0x01～0xFF
Length	16	uimsbf	数据长度
KeyType	8	uimsbf	密钥类型
KeyIdentifierLen	8	uimsbf	密钥标识长度
KeyIdentifier[]	N	uimsbf	密钥标识
KeyRulesNum	8		密钥使用规则数量
for(i=0;i<KeyRulesNum;i++)			
{			
KeyRuleType	8	uimsbf	密钥使用规则类型
KeyRuleLen	8	uimsbf	密钥使用规则长度
KeyRuleData[]	L	uimsbf	密钥使用规则数据
}			

KeyType：密钥类型，用1个字节标识该密钥的功能，密钥类型规定见表12。

KeyIdentifierLen、KeyIdentifier：密钥标识用来唯一标识该密钥，密钥唯一标识为UUID；KeyIdentifierLen表示密钥标识的长度，用1个字节表示；KeyIdentifier表示密钥标识。

KeyRulesNum：密钥使用规则数量，用1个字节表示。

KeyRuleType：密钥使用规则类型，用1个字节表示；密钥使用规则类型规定见表14。

KeyRuleLen、KeyRuleData：密钥使用规则数据；KeyRuleLen表示密钥使用规则数据的长度，用1个字节表示，KeyRuleData表示密钥使用规则数据。

表14　密钥使用规则类型

名称	类型
起始时间	0x01
截止时间	0x02
次数	0x03
时间段	0x04
累计时间段	0x05
输出规则	0x06
客户端安全等级要求	0x07
保留	0x00,0x08～0xFF

密钥使用规则规定了密钥在使用时需要遵循的约束规则，密钥使用规则包括：起始时间、截止时间、次数、时间段、累计时间段等。

起始时间：规定在该时间之后允许使用密钥，在该时间之前不允许使用密钥，该时间为UTC时间，用32位长度uimsbf类型数据表示，表示自1970年1月1日0:00:00起到该时间点的秒数。

截止时间：规定在该时间之前允许使用密钥，在该时间之后不允许使用密钥，该时间为UTC时间，用32位长度uimsbf类型数据表示，表示自1970年1月1日0:00:00起到该时间点的秒数。

次数：规定允许使用密钥的次数，用32位长度uimsbf类型数据表示。

时间段：规定从第一次使用密钥之后允许使用密钥的时间范围，用32位长度uimsbf类型数据表示，单位为秒。

累计时间段：规定从第一次使用密钥开始，累计使用密钥的时间段，每次停止使用密钥即停止计时，用32位长度uimsbf类型数据表示，单位为秒。

输出规则：规定使用内容加密密钥解密播放内容后，已解码内容数据是否允许输出到其他设备，以及允许的输出范围和方式，没有输出规则时则不限制输出方式，用8位长度uimsbf类型数据表示，具体规定见表15。

表15　输出规则

模拟输出规则（高 4 位）		数字输出规则（低 4 位）	
不限制	0000	不限制	0000
禁止输出	0001	仅允许 HDCP1.4 及以上输出	0001
保留	0010～1111	仅允许 HDCP2.2 及以上输出	0010
		禁止输出	0011
		保留	0100～1111

客户端安全等级要求：规定只允许在特定安全级别的DRM客户端进行视音频内容的解密播放，客户端安全等级要求分为软件安全级别、硬件安全级别、增强硬件安全级别，只有DRM客户端安全等级等于或者高于密钥使用规则中规定的客户端安全等级要求时，才可采用内容加密密钥进行内容的解密播放或输出，没有该密钥使用规则表示对客户端安全等级无限制。客户端安全等级要求用8位长度uimsbf类型数据表示，具体规定见表16。

表16　客户端安全等级要求

名称	类型
软件安全级别	0x01
硬件安全级别	0x02
增强硬件安全级别	0x03
保留	0x00,0x04～0xFF

7.2.7　数字签名单元

许可证的最后一个单元是数字签名单元，数字签名单元是对其前面所有单元数据进行签名，保证数据的完整性，数字签名单元数据结构见表17。

表17　数字签名单元数据结构

字段	比特数	类型	描述
Type	8	uimsbf	0xFF
Index	8	uimsbf	0x01～0xFF
Length	16	uimsbf	数据长度
Algorithm	8	uimsbf	数字签名算法

表 17（续）

字段	比特数	类型	描述
KeyIDLength	8	uimsbf	密钥标识长度
KeyID	N×8	uimsbf	密钥标识
SignatureLength	16	uimsbf	签名数据的长度
Signature[]	M×8	uimsbf	签名数据

Algorithm：采用的数字签名算法。

KeyIDLength：签名所用密钥的唯一编号长度。

KeyID：签名所用密钥的唯一编号，在当前版本中采用消息验证码作为许可证的签名，此时该唯一编号为消息验证码密钥的ID。

SignatureLength：签名数据的长度。

Signature：签名数据。

8 许可证获取协议

8.1 概述

本章规定DRM服务端与DRM客户端之间安全交换内容授权许可证的通信协议。许可证获取协议包括DRM客户端发起的许可证获取请求消息和DRM服务端回复的许可证获取响应消息。许可证获取协议消息利用URI的路径、查询信息和HTTP报文的报文体进行传输。URI固定为“http”，表示使用HTTP或HTTPS协议进行传输；URI中的域名为DRM服务端的地址，如“ri.example.com”。HTTP报文体中的消息以JSON字符串进行描述，JSON语法遵循ECMA 404。HTTP报文体中的消息样例如下：

```
{
    "object" : {
        "key1" : "string",
        "key2" : true,
        "array1" : ["elem1","elem2" ],
        "array2" : [ {
            "key1" : "value1"
        }, {
            "key2" : "value2"
        } ]
    }
}
```

8.2 许可证获取请求

DRM客户端发送许可证获取请求消息到DRM服务端请求许可证。许可证获取请求消息描述见表18。

表18 许可证请求消息描述

参数	JSON 键	值类型	描述
Type	type	string	必选
Version	version	string	必选
DeviceID	deviceID	base64_string	必选
DeviceNonce	nonce	base64_string	必选
RequestTime	requestTime	string	必选
ContentIDs	contentIDs	base64_string 数组	必选
SupportedAlgorithms	supportedAlgorithms	string 数组	必选
Extensions	extensions	对象数组	可选
CertificateChain	certificateChain	base64_string 数组	必选
Signature	signature	base64_string	必选

Type：消息类型，固定为“licenseRequest”。

Version：DRM客户端支持的最高协议版本号。当前版本Version默认为“2.0”。

DeviceID：DRM客户端标识，使用其证书的公钥摘要作为其唯一标识。

DeviceNonce：DRM客户端产生的nonce，应由随机数生成器生成。该Nonce值只应被使用一次。Nonce字符串值长度不应小于14个字节。

RequestTime：DRM客户端的当前DRM时间，该时间为UTC时间，用32位长度uimsbf类型数据的十进制字符串表示，表示自1970年1月1日0:00:00起到该时间点的秒数。

ContentIDs：DRM客户端请求许可证的内容标识信息。

SupportedAlgorithms：DRM客户端支持的密码算法，本标准固定为“KMSProfile1”。

Extensions：DRM客户端发送的扩展数据，该扩展数据为JSON对象数组，可选。本标准定义的扩展数据见表19。

表19 Extensions

参数	JSON 键	值类型	描述
DRMServerInfos	drmServerIDs	对象数组	可选
AuthenticationData	authenticationData	base64_string	可选

DRMServerInfos：包含在Extensions中的对象数组，包括DRMServerID、OCSP响应是否过期；DRM服务端可根据该数组判断是否需要发送DRM服务端证书链和OCSP响应数据，DRMServerInfos语法如下，其中ocspState为bool类型，true表示该DRMServer的OCSP响应未过期，false表示该DRMServer的OCSP响应已过期。

```
"drmServerInfos" : [
        {
            " drmServerID" : "base64_string",
             "ocspState" : bool
        },
        {
            " drmServerID" : "base64_string",
```

```
            "ocspState" : bool
        },
        …
]
```

AuthenticationData：包含在Extensions中，base64字符串，可选的用于DRM服务端向视音频内容分发服务系统请求内容的使用规则，该数据由具体的业务系统定义。

CertificateChain：DRM客户端证书链，该证书链不包括根证书，DRM客户端证书应该在证书链的第一个，后续证书必须直接证明前一个证书。

许可证获取请求消息编码格式如下：

```
{
      "type":"licenseRequest",
      "version":"2.0",
      "deviceID":"base64_string",
      "nonce":"base64_string",
      "requestTime":"string",
      "contentIDs":["base64_string","base64_string",…],
      "supportedAlgorithms":["string","string",…],
      "extensions"{
        "drmServerInfos":[…],
        "authenticationData":"base64_string"
      }
      "certificateChain":["base64_string","base64_string",…],
      "signature":"base64_string"
}
```

8.3 许可证获取响应

DRM服务端发送许可证获取响应消息到DRM客户端，该消息中包括DRM客户端请求的许可证信息；许可证响应消息描述见表20。

表20 许可证响应消息描述

参数	JSON 键	值类型	规定	
			2-pass Status=”Success”	2-pass Status≠”Success”
Type	type	string	必选	必选
Version	version	string	必选	必选
Status	status	string	必选	必选
SelectedAlgorithm	selectedAlgorithm	string	必选	必选
ResponseTime	responseTime	string	必选	必选
DeviceID	deviceID	base64_string	必选	必选
DRMServerID	drmServerID	base64_string	必选	必选

表 20（续）

参数	JSON 键	值类型	规定	
			2-pass Status=”Success”	2-pass Status≠”Success”
DeviceNonce	nonce	base64_string	必选	必选
ProtectedLicenses	protectedLicenses	对象数组	必选	-
CertificateChain	certificateChain	base64_string 数组	可选	可选
OCSPResponse	ocspResponse	base64_string	可选	可选
Signature	signature	base64_string	必选	必选

Type：消息类型，固定为“licenseResponse”。

Version：当前版本Version默认为“2.0”。

Status：指示许可证请求是否成功完成，Status规定见表21。

SelectedAlgorithm：DRM服务端选择支持的密码算法，本标准固定为“KMSProfile1”。

Version：DRM服务端与DRM客户端达成一致的协议版本号。当前版本Version默认为“2.0”，升级到高版本后应向下兼容。

ResponseTime：DRM服务端的当前DRM时间。该时间为UTC时间，用32位长度uimsbf类型数据的十进制字符串表示，表示自1970年1月1日0:00:00起到该时间点的秒数。

DeviceID：DRM客户端唯一标识，该唯一标识必须与许可证请求消息中的DRM客户端唯一标识一致。如果DRM客户端接收到的许可证响应消息中的DRM客户端标识与当前DRM客户端不一致，DRM客户端应忽略该消息。

DRMServerID：DRM服务端唯一标识，使用其证书的公钥摘要作为其唯一标识。

DeviceNonce：DRM客户端发送的随机数，该随机数必须是DRM客户端发送许可证请求消息时携带的随机数。

ProtectedLicenses：DRM服务端返回的DRM客户端请求的一个或多个许可证。许可证数组“protectedLicenses”中包含的是一个或多个许可证。ProtectedLicenses块实例如下：

```
"protectedLicenses" : [
    {
        " licenseID" : "base64_string",
         "license" : "base64_string"
     },
     {
        " licenseID" : "base64_string",
         "license" : "base64_string"
    },
     …
]
```

在“protectedLicenses”中应至少包含一个许可证。“licenseID”用Base64编码字符串表示许可证唯一标识，“license”用Base64编码字符串表示该内容对应的许可证。Base64编码方法遵循IETF RFC 2045。

CertificateChain：DRM服务端证书链。证书链是一个Base64字符串数组，每一个Base64字符串为一个DER编码的证书。该证书链不包含根证书，DRM服务端证书应排在证书链的第一个，每个后续证书应直接证明前一个证书。

OCSPResponse：DRM服务端OCSP响应。

Signature：许可证响应消息的签名。

许可证获取响应消息编码格式如下：

```
{
        "type":"licenseResponse",
        "status":"string",
        "version":"2.0",
        "selectedAlgorithm":"string",
        "responseTime":"string",
        "deviceID":"base64_string",
        "drmServerID":base64_string",
        "nonce":"base64_string",
        "protectedLicenses" : [
        {
             "licenseID" : "base64_string",
             "license" : "base64_string"
       },
       {
             "licenseID" : "base64_string",
             "license" : "base64_string"
       },
       …
       ],
       "certificateChain":["base64_string","base64_string",…],
       "ocspResponse":"base64_string",
       "signature":"base64_string"
}
```

8.4 状态信息

可能的错误状态信息包括：DRM服务端拒绝DRM客户端的请求；DRM服务端不支持当前类型的请求，如：许可证请求消息类型错误等；DRM服务端无法解析DRM客户端的请求，如：许可证请求消息编码错误等；DRM客户端没有被授权访问DRM服务端，如：DRM客户端证书过期等；DRM服务端无法验证DRM客户端的证书链，DRM服务端无法验证DRM客户端的签名、DRM客户端的时间和DRM服务端不一致、找不到DRM客户端请求的许可证对象等。当DRM客户端时间与DRM服务端不一致时，应采用该消息中的ResponseTime对DRM客户端时间进行校对，再重新进行许可证获取申请。状态信息编码见表21。

表21　状态信息编码

JSON 键	键值编码	描述
status	success	成功
	abort	DRM 服务端拒绝 DRM 客户端的请求
	typeNotSupported	DRM 服务端不支持当前类型的请求
	accessDenied	DRM 客户端没有被授权访问 DRM 服务端
	contentIDNotFound	该状态码仅会出现在许可证响应消息中，表示找不到 DRM 客户端请求的许可证对象
	malformedRequest	DRM 服务端无法解析 DRM 客户端的请求
	versionNotSupported	DRM 服务端不支持当前 DRM 客户端所使用的协议版本
	invalidCertificateChain	DRM 服务端无法验证 DRM 客户端的证书链
	signatureError	DRM 服务端无法验证 DRM 客户端的签名
	deviceTimeError	DRM 客户端的时间和 DRM 服务端不一致
	drmClientSecurityLevelError	DRM 客户端安全等级低于要求的安全等级

8.5　消息签名机制

DRM 客户端产生签名为空字符串的 LicenseRequest JSON String，对该 string 计算签名，将签名结果填入 LicenseRequest JSON String。

DRM 服务端产生签名为空字符串的 LicenseResponse JSON String，对该 string 计算签名，将签名结果填入 LicenseResponse JSON String。

9　DRM 服务端

9.1　概述

DRM 服务端包括：内容加密、密钥管理、密钥网关、内容授权等核心模块。内容加密负责直播或点播内容的加密封装；密钥管理负责接收内容加密的内容加密密钥信息并进行安全存储和发布；密钥网关负责接收各密钥管理同步的内容加密密钥，并接收内容授权的密钥查询；内容授权从密钥网关查询内容加密密钥，封装成许可证发送给 DRM 客户端。DRM 服务端框架如图 11 所示。

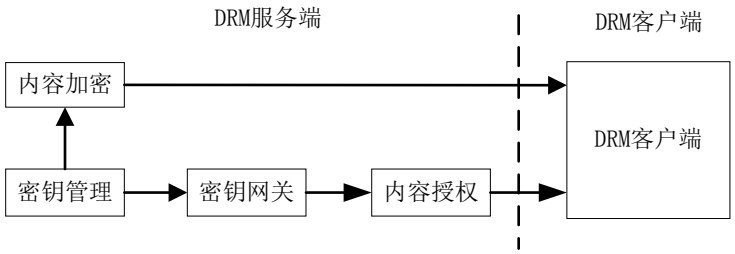

图11　DRM 服务端框架

本章规定密钥管理与密钥网关之间的密钥同步协议，以及密钥网关与内容授权之间的密钥查询协议。密钥同步和密钥查询协议采用 HTTP/HTTPS 协议，POST（JSON）接口，证书链全部采用 DER 编码格式。

9.2 密钥同步协议

9.2.1 概述

密钥同步是指内容提供者的密钥管理将直播或点播内容加密密钥同步到密钥网关。

密钥同步包括密钥同步请求和密钥同步响应两条消息。

9.2.2 密钥同步请求

密钥同步请求消息由密钥管理发起，密钥网关验证并响应，密钥同步请求消息包括：版本号、密钥管理唯一标识、随机数、内容 ID、内容加密密钥、密钥管理证书链、数字签名等。密钥同步请求消息见表 22。

内容加密密钥由密钥网关的公钥加密，密钥网关接收后不解密，只支持将此内容加密密钥转换为使用 DRM 客户端的公钥进行加密。

请求中的数字签名使用密钥管理的私钥生成。

表22 密钥同步请求消息

JSON 键	值类型	描述
type	string	必选
version	String	必选
kmsID	base64_string	必选
nonce	base64_string	必选
selectedAlgorithm	string	必选
contentInfos	对象数组	必选
contentID	base64_string	必选
ceks	对象数组	必选
cekID	base64_string	必选
encCEK	base64_string	必选
extension	base64_string	可选
startTime	string	可选
endTime	string	可选
certificateChain	base64_string 数组	必选
signature	base64_string	必选

type：消息类型，固定为“keySyncRequest”。

version：当前消息协议版本号。当前版本 Version 默认为“1.0”。

kmsID：密钥管理标识，这里指的是内容提供者的密钥管理的唯一标识。

nonce：消息发送方产生的 nonce，应由随机数生成器生成。

selectedAlgorithm：固定为“KMSProfile1”。

contentInfos：内容 ID 和内容加密密钥对象的数组。

contentID：内容标识符。

ceks：内容加密密钥对象数组，包含内容加密密钥标识符和内容加密密钥。

cekID：内容加密密钥标识符。

encCEK：用密钥网关证书公钥加密的内容加密密钥。

extension：扩展信息。

startTime：密钥起始时间，32 位 UTC 时间的十进制字符串。

endTime：密钥截止时间，32 位 UTC 时间的十进制字符串。

certificateChain：密钥管理证书链，该证书链不包括根证书。

signature：消息的签名。

密钥同步请求消息编码格式如下：

```
{
      "type":" keySyncRequest ",
      "version":"1.0",
      "kmsID":"base64_string",
      "nonce":"base64_string",
      "selectedAlgorithm":"string",
      "contentInfos":[
          {
            "contentID":"base64_string",
            "ceks":[
                  {
                    "cekID":"base64_string",
                    "encCEK":"base64_string",
                    "startTime":"string",
                    "endTime":"string"
                  },…],
            "contentRules":"base64_string"
          },
          …],
      "certificateChain":["base64_string","base64_string",…],
      "signature":"base64_string"
}
```

9.2.3 密钥同步响应

密钥同步响应消息包括：版本号、密钥网关唯一标识、随机数、状态信息、密钥网关证书链、数字签名等。密钥同步响应消息见表 23。

状态信息为密钥同步处理的状态，包括：同步成功、时间超时、无法解密内容加密密钥、内容加密系统证书不合法、数字签名不正确、未知状态等。

数字签名为密钥网关使用私钥计算的消息签名。

表23 密钥同步响应消息

JSON 键	值类型	描述
type	string	必选
version	string	必选
keyGateWayID	base64_string	必选

表 23（续）

JSON 键	值类型	描述
nonce	base64_string	必选
status	string	必选
selectedAlgorithm	string	必选
certificateChain	base64_string 数组	必选
signature	base64_string	必选

type：消息类型，固定为“keySyncResponse”。

version：当前消息协议版本号。当前版本 Version 默认为“1.0”。

keyGateWayID：密钥网关的唯一标识。

nonce：消息发送方产生的 nonce，应与密钥同步请求消息中的一致。

status：反馈的状态信息，密钥同步响应状态信息规定见表 24。

表24　密钥同步响应状态信息

状态值	描述
success	同步成功
doNotSupportSelectedAlgorithm	不支持请求的算法
certificationInvalid	密钥管理证书不合法
signatureInvalid	密钥同步请求消息数字签名不正确
unknownError	未知错误

selectedAlgorithm：固定为“KMSProfile1”。

certificateChain：密钥网关证书链，该证书链不包括根证书。

signature：消息的签名。

密钥同步响应消息编码格式如下：

```
{
    "type":" keySyncResponse ",
    "version":"1.0",
    " keyGateWayID ":"base64_string",
    "nonce":"base64_string",
    "status":"string",
    "selectedAlgorithm":"string",
    "certificateChain":["base64_string","base64_string",…],
    "signature":"base64_string"
}
```

9.3　密钥查询协议

9.3.1　概述

密钥查询是指内容授权从密钥网关查询客户端需要的内容加密密钥。

密钥查询包括密钥查询请求和密钥查询响应两条消息。

9.3.2 密钥查询请求

密钥查询请求消息由内容授权发送到密钥网关。密钥查询请求消息包括：版本号、DRM 服务端唯一标识、随机数、内容 ID、DRM 客户端证书链、内容授权证书链、数字签名。密钥查询请求消息见表 25。

表25 密钥查询请求消息

JSON 键	值类型	描述
type	string	必选
version	string	必选
drmServerID	base64_string	必选
nonce	base64_string	必选
selectedAlgorithm	string	必选
contentIDs	对象数组	必选
drmClientCertificate	base64_string	必选
certificateChain	base64_string 数组	必选
signature	base64_string	必选

type：消息类型，固定为“keyRequest”。
version：当前消息协议版本号。当前版本 Version 默认为“1.0”。
drmServerID：内容授权的唯一标识。（证书公钥的摘要）
nonce：消息发送方产生的 nonce，应由随机数生成器生成。
contentIDs：内容 ID 对象数组，包含内容 ID 和可选的起始时间、截止时间。
selectedAlgorithm：固定为“KMSProfile1”。
drmClientCertificate：DRM 客户端证书。
certificateChain：内容授权证书链，该证书链不包括根证书。
signature：消息的签名。
密钥查询请求消息编码格式如下：

```
{
    "type":" keyRequest ",
    "version":"1.0",
    "drmServerID":"base64_string",
    "nonce":"base64_string",
    "selectedAlgorithm":"string",
    "contentIDs":[
            {
                "contentID":"base64_string",
                "startTime":"string",
                "endTime":"string"
              },
    …],
    "drmClientCertificate":"base64_string",
```

```
    "certificateChain":["base64_string","base64_string",…],
    "signature":"base64_string"
}
```

9.3.3 密钥查询响应

密钥查询响应消息由密钥网关发送到内容授权。密钥查询响应消息包括：版本号、密钥网关唯一标识、随机数、内容 ID、状态信息、会话密钥、内容加密密钥、密钥网关证书链、数字签名等。密钥查询响应消息见表 26。

表26 密钥查询响应消息

JSON 键	值类型	描述
type	string	必选
version	string	必选
keyGateWayID	base64_string	必选
nonce	base64_string	必选
status	string	必选
selectedAlgorithm	string	status=”success”时必选
cekInfos	对象数组	status=”success”时必选
contentID	base64_string	status=”success”时必选
sessionKeyID	base64_string	status=”success”时必选
encSessionKey	base64_string	status=”success”时必选
encCEKs	对象数组	status=”success”时必选
cekID	base64_string	status=”success”时必选
encCEK	base64_string	status=”success”时必选
contentRules	base64_string	可选
certificateChain	base64_string 数组	必选
signature	base64_string	必选

type：消息类型，固定为“keyResponse”。

version：当前消息协议版本号。当前版本 Version 默认为“1.0”。

keyGateWayID：密钥网关的唯一标识。

nonce：消息发送方产生的 nonce，应与密钥查询请求消息的 nonce 一致。

status：反馈的状态信息，包括：查询成功、时间不合法、找不到该内容 ID、DRM 客户端证书不合法、内容授权证书不合法、内容授权不在白名单中、未知错误等，见表 27。

selectedAlgorithm：固定为“KMSProfile1”。

表27 密钥查询响应状态信息

状态值	描述
success	查询成功
doNotSupportSelectedAlgorithm	不支持请求的算法
contentIDInvalid	找不到该内容 ID

表 27（续）

状态值	描述
deviceCertInvalid	DRM 客户端证书不合法
drmServerCertInvalid	内容授权证书不合法
drmServerIDInvalid	内容授权不在白名单中
signatureInvalid	密钥同步请求消息数字签名不正确
unknownError	未知错误

cekInfos：内容加密密钥对象数组。

contentID：内容 ID。

sessionKeyID：会话密钥标识符。

encSessionKey：会话密钥为 DRM 客户端公钥加密的随机密钥，即内容授权许可证中的会话密钥。会话密钥按照内容授权许可证格式中的要求进行加密。

encCEKs：内容加密密钥数组。

cekID：内容加密密钥标识符。

encCEK：内容加密密钥为会话密钥加密的内容加密密钥，内容加密密钥由会话密钥按照内容授权许可证格式的要求进行加密。

contentRules：内容使用规则。

certificateChain：密钥网关证书链，该证书链不包括根证书。

signature：消息的签名。

密钥查询响应消息编码格式如下：

```
{
    "type":" keyResponse ",
    "version":"1.0",
    "keyGateWayID":"base64_string",
    "nonce":"base64_string",
    "status":"string",
    "selectedAlgorithm":"string",
    "cekInfos":[
        {
          "contentID":"base64_string",
          "sessionKeyID":"base64_string",
          "encSessionKey":"base64_string",
          "encCEKs":[
                {
                  "cekID":"base64_string",
                  "encCEK":"base64_string"
                },…],
          "contentRules":"base64_string"
        },
        …],
```

```
    "certificateChain":["base64_string","base64_string",…],
    "signature":"base64_string"
}
```

10 DRM 客户端

10.1 概述

DRM 客户端是设备中的可信实体，负责执行与 DRM 内容相关的许可和限制。DRM 客户端包括：DRM 应用模块、DRM 功能模块、DRM 客户端运行环境，以及 DRM 功能接口和 DRM 客户端运行环境接口；DRM 客户端框架如图 12 所示。

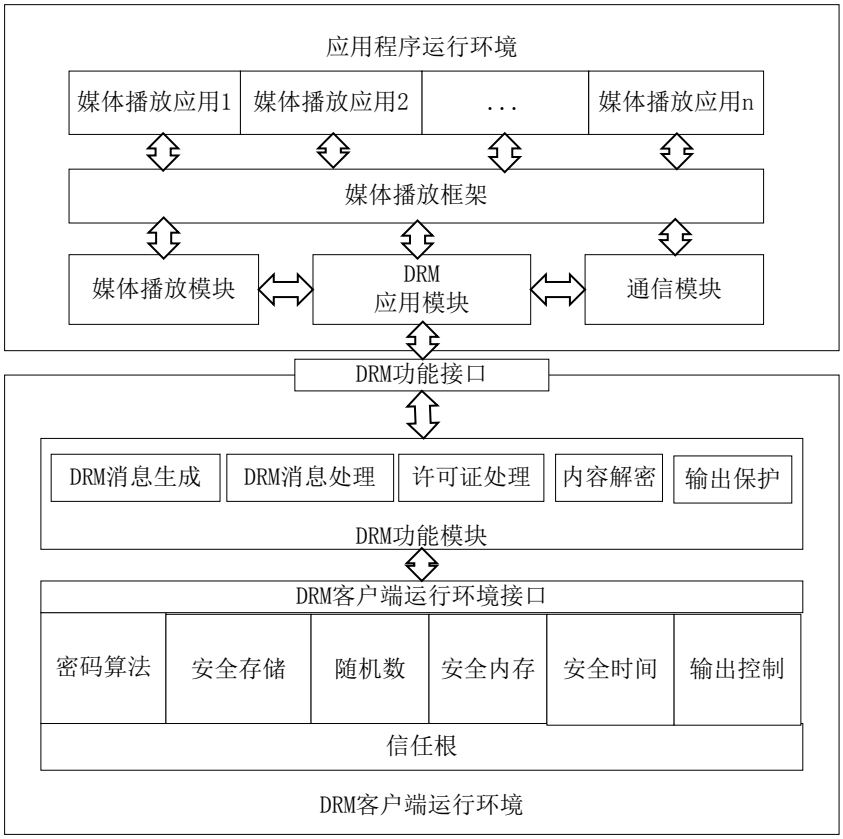

图12 DRM 客户端框架

DRM 应用模块是负责 DRM 功能与媒体播放等应用集成的模块，DRM 客户端应用模块与媒体播放框架集成，实现媒体播放应用对 DRM 的支持，通过调用 DRM 功能接口实现 DRM 功能。

DRM 功能模块为 DRM 应用模块提供 DRM 功能。DRM 功能模块运行在 DRM 客户端运行环境中，通过调用 DRM 客户端运行环境接口实现 DRM 客户端的核心功能。

DRM 客户端运行环境负责为 DRM 功能模块提供安全能力，为 DRM 客户端信任与安全体系建立、DRM 客户端核心功能等的运行提供基础的安全保障。

本标准定义的 DRM 客户端分为三个安全级别：软件安全级别、硬件安全级别、增强硬件安全级别。软件安全级别是指部分或全部 DRM 客户端运行环境基于软件安全机制实现；硬件安全级别是指 DRM 客户端运行环境基于硬件安全机制实现；增强硬件安全级别是指在硬件安全级别的基础上，DRM 客户端安全

运行环境应具备侧信道攻击防御、取证水印等功能。

10.2 DRM 应用模块

DRM 应用模块运行在应用程序运行环境中，是媒体相关应用集成 DRM 客户端功能的逻辑模块。DRM 应用模块通过 DRM 功能接口调用 DRM 功能模块实现对内容的保护。

10.3 DRM 功能模块

DRM 功能模块应在 DRM 客户端运行环境中执行，支持 DRM 消息生成、DRM 消息处理、许可证处理、内容解密、输出保护、DRM 客户端私钥及证书导入功能。

DRM 客户端私钥、内容加密密钥、内容授权许可证等应在 DRM 客户端运行环境中安全存储，且仅在 DRM 客户端运行环境中使用密钥，进行密钥相关计算的操作。

10.4 DRM 客户端运行环境

DRM 客户端运行环境支持信任根、密码算法、安全存储、随机数、安全内存、安全时间、输出控制等功能；在芯片或设备出厂时如果未置入 DRM 客户端证书和私钥，DRM 客户端运行环境应具备 DRM 客户端证书和私钥安全置入功能。

DRM 客户端运行环境应具有保护其自身完整性和 DRM 功能模块完整性的能力，在 DRM 功能模块运行之前应能够检查 DRM 功能模块的完整性，防止 DRM 核心功能被篡改。

DRM 功能模块运行在 DRM 客户端运行环境中，该运行环境是逻辑上的，实际实施过程中可以是基于芯片硬件的安全运行环境（如 TEE 等），可以是基于软件的安全运行环境（如 WhiteBox 等），也可以是软件和硬件混合构成的安全运行环境。

10.5 DRM 功能接口

DRM 功能接口是运行在 DRM 客户端运行环境中的 DRM 功能模块对外提供的用于与 DRM 应用模块集成的接口，接口定义见附录 C。

10.6 DRM 客户端运行环境接口

DRM 客户端运行环境接口是 DRM 客户端运行环境为 DRM 功能模块提供的 DRM 安全能力接口，包括密码算法、安全存储、随机数、安全内存、安全时间、输出控制接口，接口定义见附录 D。

附 录 A
（规范性附录）
数字证书格式、在线证书认证协议及证书撤销列表格式

A.1 概述

本附录规定本标准数字证书格式、在线证书认证协议消息格式、以及证书撤销列表格式。
本标准数字证书格式应符合GB/T 20518—2018的规定。

A.2 数字证书格式

A.2.1 根CA证书

根CA证书是一个自签名的CA证书，DRM服务端和DRM客户端以该证书为信任根来验证其他证书和吊销列表。根CA证书格式见表A.1。

表A.1 根 CA 证书格式

字段	说明
Version	版本号，当前版本为 v3，整型值 2
SerialNumber	序列号。4字节随机数
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与主体域相同
Validity	有效期，自签名之日起50年内有效
Subject	主体，包含属性： ——国家/地区名称 (countryName)（例如"CN"）； ——单位名称 (oranizationName)（例如"CDTA"）； ——常用名 (commonName) （例如"Root CA"）
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，为必选、关键字段。CA字段被设置为TRUE，不使用pathLenConstraint字段
CRLDistributionPoints extension	CRL分发点扩展，不使用
KeyUsage extension	密钥使用扩展，为必选、 关键字段。只设置密钥使用位keyCertSign和cRLSign
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，不使用

表 A.1（续）

字段	说明
SubjectKeyIdentifier extension	主体密钥标识扩展，为必选、非关键字段。密钥标识符方法遵循IETF RFC 3280，用subjectPublicKey字段中不包括标记、 长度和未使用的位计算的SM3哈希值
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，不使用
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，不使用
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501
SignatureValue	签名值

A.2.2 DRM服务端CA证书

DRM服务端CA证书由根CA签发,DRM服务端保存该证书。DRM服务端可将该证书发给DRM客户端作为DRM服务端证书链的一部分。DRM服务端CA证书见表A.2。

表A.2 DRM 服务端 CA 证书格式

字段	说明
Version	版本号，当前版本为v3，整型值2
SerialNumber	序列号。4字节随机数
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与根CA主体字段相同
Validity	有效期，截止日期与根CA截止相同,开始日期为证书的生成时间
Subject	主体，包含属性： ——国家/地区名称 (countryName)（例如"CN"）； ——单位名称 (organizationName)（例如"CDTA"）； ——常用名 (commonName) （例如"Licese Issuer CA 1"）
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，必选、关键。CA字段被设置为 TRUE，pathLenConstraint字段设置为零
CRLDistributionPoints extension	CRL分发点扩展，不使用
KeyUsage extension	密钥使用扩展，必选、关键。只设置密钥使用位keyCertSign和cRLSign
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，必选、非关键。该扩展只包含keyIdentifier字段，具有和相应根CA证书的subjectKeyIdentifier扩展值相同的值
SubjectKeyIdentifier extension	主体密钥标识扩展，必选、非关键。密钥标识符方法遵循IETF RFC 3280，用subjectPublicKey字段中不包括标记、长度和未使用的位计算的SM3哈希值

表 A.2（续）

字段	说明
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，不使用
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，不使用
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501
SignatureValue	签名值

A.2.3 DRM服务端证书

DRM服务端证书由DRM服务端CA签发，该证书是DRM服务端证书链的第一个证书。DRM服务端证书格式见表A.3。

表A.3 DRM 服务端证书格式

字段	说明
Version	版本号，当前版本为v3，整型值2
SerialNumber	序列号，4字节随机数
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与DRM服务端CA证书的主体字段相同
Validity	有效期，有效期为签发日期起5年内，但不应晚于相应DRM服务端CA证书的证书有效日期
Subject	主体， 强制属性： 组织名称 (organizationName)（如"示例名称"）； 可选属性： 由DRM服务端提供
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，不使用
CRLDistributionPoints extension	CRL分发点扩展，不使用
KeyUsage extension	密钥使用扩展，必选、关键。只设置密钥使用位digitalSignature
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，必选、非关键。该扩展只包含keyIdentifier字段，具有和相应DRM服务端CA证书的subjectKeyIdentifier扩展值相同的值
SubjectKeyIdentifier extension	主体密钥标识扩展，不使用
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，必选、关键。该扩展包含cdrm-kp-licenseIssuer对象标识符(1.2.156.112560.7)
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，不使用

表 A.3（续）

字段	说明
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501
SignatureValue	签名值

A.2.4 OCSP响应器证书

OCSP响应器证书由DRM服务端CA签发，OCSP响应器保存该证书。该证书同OCSP响应一同由DRM服务端发送给DRM客户端。OCSP响应器证书格式见表A.4。

表A.4 OCSP 响应器证书格式

字段	说明
Version	版本号，当前版本为v3，整型值2
SerialNumber	序列号。4字节随机数
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与相应的DRM服务端CA主体字段相同
Validity	有效期，3个月有效期，但应在DRM服务端CA证书有效期内
Subject	主体，包含属性： ——国家/地区名称（countryName）（例如"CN"）； ——组织名称（organizationName）（例如"CDTA"）； ——常用名（commonName）（例如"OCSP Responder 1"）
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，不使用
CRLDistributionPoints extension	CRL分发点扩展，不使用
KeyUsage extension	密钥使用扩展，必选、关键。只设置密钥使用法 digitalSignature
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，必选、非关键。该扩展只包含 keyIdentifier 字段，具有和相应的DRM服务端CA 证书的 subjectKeyIdentifier 扩展值相同的值
SubjectKeyIdentifier extension	主体密钥标识扩展，必选、非关键。密钥标识符方法使用，见IETF RFC 3279，用subjectPublicKey字段中不包括标记、长度和未使用的位计算的SM3哈希值
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，必选、关键。该扩展包含id-kp-OCSPSigning 对象标识符
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，必选、非关键。扩展值为 NULL
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501
SignatureValue	签名值

A.2.5 密钥管理证书

密钥管理证书由DRM服务端CA签发，该证书用来标识密钥管理系统的身份。密钥管理证书格式见表A.5。

表A.5　密钥管理证书格式

字段	说明
Version	版本号，当前版本为v3，整型值2
SerialNumber	序列号，4字节随机数
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与DRM服务端CA证书的主体字段相同
Validity	有效期，有效期为签发日期起5年内，但不应晚于相应DRM服务端CA证书的证书有效日期
Subject	主体， 强制属性： 组织名称（organizationName）（如"示例名称"）； 可选属性： 由DRM服务端提供
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，不使用
CRLDistributionPoints extension	CRL分发点扩展，不使用
KeyUsage extension	密钥使用扩展，必选、关键。只设置密钥使用位digitalSignature
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，必选、非关键。该扩展只包含keyIdentifier字段，具有和相应DRM服务端CA证书的subjectKeyIdentifier扩展值相同的值
SubjectKeyIdentifier extension	主体密钥标识扩展，不使用
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，必选、关键。该扩展包含 cdrm-kp-licenseIssuer 对象标识符(1.2.156.112560.23)
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，不使用
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501
SignatureValue	签名值

A.2.6　密钥网关证书

密钥网关证书由DRM服务端CA签发，该证书用来标识密钥网关系统的身份。密钥网关证书格式见表A.6。

表A.6　密钥网关证书格式

字段	说明
Version	版本号，当前版本为v3，整型值2
SerialNumber	序列号，4字节随机数
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与DRM服务端CA证书的主体字段相同
Validity	有效期，有效期为签发日期起5年内，但不应晚于相应DRM服务端CA证书的证书有效日期
Subject	主体， 强制属性： 组织名称（organizationName）（如"示例名称"）； 可选属性： 由DRM服务端提供
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，不使用
CRLDistributionPoints extension	CRL分发点扩展，不使用
KeyUsage extension	密钥使用扩展，必选、关键。只设置密钥使用位digitalSignature
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，必选、非关键。该扩展只包含keyIdentifier字段，具有和相应DRM服务端CA证书的subjectKeyIdentifier扩展值相同的值
SubjectKeyIdentifier extension	主体密钥标识扩展，不使用
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，必选、关键。该扩展包含 cdrm-kp-licenseIssuer 对象标识符(1.2.156.112560.24)
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，不使用。
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501
SignatureValue	签名值

A.2.7　内容加密证书

内容加密证书由DRM服务端CA签发，该证书用来标识内容加密系统的身份。内容加密证书格式见表A.7。

表A.7　内容加密证书格式

字段	说明
Version	版本号，当前版本为v3，整型值2
SerialNumber	序列号，4字节随机数
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与DRM服务端CA证书的主体字段相同
Validity	有效期，有效期为签发日期起5年内，但不应晚于相应DRM服务端CA证书的证书有效日期
Subject	主体， 强制属性： 组织名称（organizationName）（如"示例名称"）； 可选属性： DRM服务端提供
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，不使用
CRLDistributionPoints extension	CRL分发点扩展，不使用
KeyUsage extension	密钥使用扩展，必选、关键。只设置密钥使用位digitalSignature
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，必选、非关键。该扩展只包含keyIdentifier字段，具有和相应DRM服务端CA证书的subjectKeyIdentifier扩展值相同的值
SubjectKeyIdentifier extension	主体密钥标识扩展，不使用
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，必选、关键。该扩展包含 cdrm-kp-licenseIssuer 对象标识符(1.2.156.112560.22)
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，不使用
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501
SignatureValue	签名值

A.2.8　DRM客户端CA证书

DRM客户端CA证书由根CA签发,DRM客户端应保存DRM客户端CA证书。DRM服务端和DRM客户端交互时，DRM客户端将该证书发给DRM服务端作为DRM客户端证书链的一部分。DRM客户端CA证书格式见表A.8。

表A.8 DRM客户端CA证书格式

字段	说明
Version	版本号，当前版本为v3，整型值2
SerialNumber	序列号。4字节随机数
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与根CA主体字段相同
Validity	有效期，到期日期与根CA到期日期相同,开始日期为证书的生成时间
Subject	主体，包含属性： ——国家/地区名称（countryName）（例如"CN"）； ——组织名称（organizationName）（例如"CDTA"）； ——常用名（commonName）（例如"Device CA 1"）
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，必选、关键。cA字段被设置为 TRUE。PathLenConstraint 字段设置为零
CRLDistributionPoints extension	CRL分发点扩展，不使用
KeyUsage extension	密钥使用扩展，必选、关键。对于 证书中心拥有的设备 CA，只设置密钥使用位keyCertSign和cRLSign
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，必选、非关键。该扩展只包含 keyIdentifier 字段，具有和相应的根CA证书的 subjectKeyIdentifier 扩展值相同的值
SubjectKeyIdentifier extension	主体密钥标识扩展，必选、非关键。密钥标识符方法遵循IETF RFC 3279，用subjectPublicKey字段中不包括标记、长度和未使用的位计算的SM3哈希值
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，不使用
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，不使用
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501。
SignatureValue	签名值

A.2.9 DRM客户端证书

DRM客户端证书由DRM客户端CA签发。DRM客户端在接到请求时将DRM客户端证书发送给DRM服务端，作为DRM客户端证书链的第一个证书。DRM客户端证书格式见表A.9。

表A.9 DRM客户端证书格式

字段	说明
Version	版本号，当前版本为v3，整型值2
SerialNumber	序列号，证书中心已颁发的设备证书内唯一的非负整数，用DER编码的序列号字段的结构规定如下： 字节 1～2：CA 标识符。第一个字节必须不是 00h 或者其最高有效位不为零； 字节 3～6：证书批次号,设备CA内唯一； 字节7～20：用证书公钥和当前日期和时间计算的SM3哈希值的最低14字节。 序列号可用于在设备CRL中吊销一批密钥和证书
Signature	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016
Issuer	颁发者，与颁发设备 CA 证书的主体字段相同
Validity	有效期，有效期为 20年，但不应晚于相应设备 CA 证书的有效期
Subject	主体，强制属性： ——单位名称(organizationName)由证书中心给特定的设备制造商指定的； ——序列号(serialNumber)由证书中心分配，通过计算公钥的SM3哈希值得到； ——常用名(commonName)由三部分构成： • 第一部分表示使用该证书的设备的安全级别： “SW-” 标志实现了软件安全级别的客户端； “HW-” 标志实现了硬件安全级别的客户端； “EH-” 标志实现了增强硬件安全级别的客户端。 • 第二部分为设备商通过测试后获得的备案编号。 • 第三部分由设备商或服务商定义，可为空。 可选属性： 由设备制造商指定
SubjectPublicKeyInfo	主体公钥信息，包含： ——算法标识 id-ecPublicKey OID 1.2.840.10045.2.1； ——SM2曲线标识 OID 1.2.156.10197.1.301； ——SM2公钥
IssuerUniqueID	颁发者唯一标识符，不使用
SubjectUniqueID	主体唯一标识符，不使用
BasicConstraints extension	基本约束扩展，不使用
CRLDistributionPoints extension	CRL分发点扩展： a) 由证书中心管理的设备CA颁发的证书：不使用； b) 由辅助设备CA颁发的证书：必选、非关键。 该扩展只包含一个cRLIssuer 字段用来定义设备 CRL 颁发者的可识别名称
KeyUsage extension	密钥使用扩展，必选、 关键。只设置密钥使用位 digitalSignature 和 keyEncipherment
AuthorityKeyIdentifier extension	颁发者密钥标识扩展，必选、非关键。该扩展只包含 keyIdentifier 字段，具有与相应设备CA证书的 subjectKeyIdentifier 扩展值相同的值
SubjectKeyIdentifier extension	主体密钥标识扩展，不使用

表 A.9（续）

字段	说明
CertificatePolicies extension	证书策略扩展，不使用
ExtKeyUsage extension	扩展密钥使用扩展，必选、关键。该扩展包含 cdrm-kp-drmAgent 对象标识符 (1.2.156.112560.8)
Id pkix ocsp nocheck extension	不查OCSP响应器证书扩展，不使用
SignatureAlgorithmId	签名算法标识，SM3 with SM2, OID 1.2.156.10197.1.501
SignatureValue	签名值

A.3 OCSP协议

OCSP服务遵循IETF RFC 2560的规定。OCSP协议消息传输使用HTTP协议，OCSP服务请求由DRM服务端向OCSP响应器发起，OCSP响应器回复后，由DRM服务端将该响应在许可证获取协议中发送给DRM客户端，DRM客户端基于该OCSP响应验证DRM服务端的合法性。

OCSP请求消息格式见表A.10。

表A.10 OCSP 请求消息格式

字段	说明
Version	版本，v2（整数值 1）
RequestorName	请求者名，可选；可包含在请求中，但OCSP响应器将忽略请求者名
RequestList	请求列表，应包含一个请求列表
Request	请求，应包含一个列表
ReqCert	请求证书字段
Hash-Algrithm	哈希算法，SM3
IssuerName Hash	颁发者DN值的哈希值，该哈希值不应被截断
IssuerKey Hash	颁发者公钥的哈希值，该哈希值将不应被截断
SerialNumber	证书序列号
SingleRequest-extension	单个请求的扩展，不使用
RequestExtensions	请求(列表)的扩展，不应使用除以下定义以外的扩展
OptionalSignature	可选签名，可能包含在请求中，但OCSP响应器将其忽略

OCSP响应消息格式见表A.11。

表A.11 OCSP 响应消息格式

字段	说明
ResponseStatus	响应状态，成功 - successful，异常请求 - malformedRequest，内部错 - internalError，等一会再试 - tryLater，必须签名 - sigRequired，未经正常授权 - unauthorized as appropriate
ResponseBytes	响应字段
ResponseType	响应类型，OCSP 基本响应：id-pkix-ocsp-basic
Response	响应，八位字节串包含BasicOcspResponse
BasicOcspResponse	基本OCSP 响应
TbsResponseData	将被签名的响应数据
Version	版本，v2（整数值1）
ResponderID	响应器标识符，应使用密钥（不是名称）哈希，其值和 OCSP 响应器证书的扩展域 subjectKeyIdentifier 的值相同
ProducedAt	生成时间，生成此 OCSP 响应的时间
Responses	响应列表
CertID	证书标识
HashAlgorithm	哈希算法，SM3，哈希值不应被截断
IssuerName-Hash	颁发者名哈希，颁发者DN 的哈希值
IssuerKeyHash	颁发者公钥哈希，颁发者公钥的哈希值
SerialNumber	证书序列号
CertStatus	证书状态，可选择值：正常 - good、吊销 - revoked、未知 - unknown
ThisUpdate	本次修改时间，取自于 CRL 的 thisUpdate 字段值
NextUpdate	下次修改时间，取自于 CRL 的 nextUpdate 字段值
SingleExtensions	单个响应扩展，不使用
SignatureAlgorithm	签名算法，SM3 with SM2, OID 1.2.156.10197.1.501，见GB/T 32918.2—2016。
Signature	签名值，根据签名算法计算的签名值
Certs	证书列表，OCSP 响应器证书由OCSP 响应器通过每个 OCSP 响应提供给DRM服务端。但是在把响应传递给设备前，此证书可能从 OCSP 响应中去除。DRM服务端CA 和根 CA 的证书不需要提供，它们应在设备置入证书时已经预先置入

A.4 证书撤销列表

证书撤销列表包括：DRM客户端CA证书撤销列表、DRM客户端证书撤销列表、DRM服务端证书撤销列表。

DRM服务端根据DRM客户端CA证书撤销列表、DRM客户端证书撤销列表验证DRM客户端证书链的状态。

DRM服务端OCSP响应器基于DRM服务端证书撤销列表给出DRM服务端证书OCSP响应消息。证书撤销列表遵循IETF RFC 3280。证书撤销列表都为DER编码，证书颁发者和拥有者的属性采用UTF8String编码。证书撤销列表语法如下：

```
CertificateList  ::=  SEQUENCE  {
    tbsCertList            TBSCertList,
    signatureAlgorithm     AlgorithmIdentifier,
```

```
      signatureValue         BIT STRING  }

TBSCertList  ::=  SEQUENCE  {
      version                Version OPTIONAL,
                                   -- if present, MUST be v2
      signature              AlgorithmIdentifier,
      issuer                 Name,
      thisUpdate             Time,
      nextUpdate             Time OPTIONAL,
      revokedCertificates    SEQUENCE OF SEQUENCE  {
           userCertificate         CertificateSerialNumber,
           revocationDate          Time,
           crlEntryExtensions      Extensions OPTIONAL
                                         -- if present, MUST be v2
                              }  OPTIONAL,
      crlExtensions          [0]  EXPLICIT Extensions OPTIONAL
                                         -- if present, MUST be v2
                              }
```

附 录 B
（规范性附录）
密码算法

本附录规定视音频内容分发数字版权管理系统支持的密码算法。

本标准内容授权许可证格式、许可证获取协议、密钥同步协议、密钥查询协议中涉及到的密码算法定义为“KeyProfile1”，具体规定如下：

a) 视音频内容采用SM4算法加密，SM4算法见GB/T 32097—2016；加密模式为CBC模式，见GB/T 17964—2008；

b) DRM服务端生成会话密钥，用该会话密钥加密内容加密密钥，加密算法采用SM4算法，加密模式为ECB模式，加密模式见GB/T 17964—2008；

c) DRM服务端用DRM客户端公钥加密会话密钥；加密算法使用SM2公钥加密算法，见GB/T 32918.4—2016；

d) 如果内容加密密钥需要同步到密钥网关，则采用密钥网关公钥加密，加密算法使用SM2公钥加密算法，见GB/T 32918.4—2016；

e) DRM服务端将加密后的会话密钥、加密后的内容加密密钥封装在许可证中发送给DRM客户端；许可证中的消息验证码算法见IETF RFC 2104，其中HMAC算法用到的摘要算法采用GB/T 32905—2016定义的SM3算法，密钥为32字节随机数。

许可证密钥单元中的密钥算法用1个字节标识，字节的高4位是密码算法的类别编码，低4位是密码算法的算法编号。密码算法规定见表B.1。

表 B.1 密码算法规定

算法类别	类别编号	算法名称	算法编号
散列算法 （HashAlgorithm）	0000	HashAlgorithm:SHA-1	0000
		HashAlgorithm:SHA-256	0001
		HashAlgorithm:SM3-256	0010
		保留	0100～1111
公钥加密算法 (PublicKeyAlgorithm)	0001	PublicKeyAlgorithm: RSA-1024	0000
		PublicKeyAlgorithm:RSA-2048	0001
		PublicKeyAlgorithm:SM2-256	0010
		RSAES-KEM-KWS	1000
		保留	0011～1111
分组密码算法 （BlockCipherAlgorithm）	0010	BlockCipherAlgorithm:AES-128-128	0000
		BlockCipherAlgorithm:3DES-64-112	0001
		BlockCipherAlgorithm:SM4-128	0010
		BlockCipherAlgorithm:AES-WRAP	1000
		保留	0101～1111
流密码算法 （StreamCipherAlgorithm）	0011	StreamingCipherAlgorithm:RC4	0000
		保留	0001～1111

表 B.1（续）

算法类别	类别编号	算法名称	算法编号
签名算法 （SignatureAlgorithm）	0100	SignatureAlgorithm:RSA-SHA1-1024	0000
		SignatureAlgorithm:RSA-SHA1-2048	0001
		PublicKeyAlgorithm:SM2-256	0010
		HMAC-SM3	0011
		HMAC-SHA1	1000
		SHA1-RSA_PKCS1_PSS_PADDING	1001
		保留	0011～1111
保留	0110～1111	保留	0000～1111

附 录 C
（规范性附录）
DRM 客户端功能接口

C.1 常量定义

C.1.1 授权状态CDRMC_Rights_Status

原型：typedef enum {

```
CDRMC_RIGHTS_VALID          = 0x00,
CDRMC_RIGHTS_INVALID        = 0x01,
CDRMC_RIGHTS_NOT_ACQUIRED = 0x02,
CDRMC_RIGHTS_EXPIRED        = 0x03,
```

} CDRMC_Rights_Status;

描述：授权状态。

成员：

CDRMC_RIGHTS_VALID：权限有效；

CDRMC_RIGHTS_INVALID：权限无效；

CDRMC_RIGHTS_NOT_ACQUIRED：没有权限；

CDRMC_RIGHTS_NOT_EXPIRED：权限过期。

C.1.2 对称加解密算法及模式CDRMC_Symmetric_Crypto_Algorithm

原型：typedef enum {

```
CDRMC_ALG_SM4_CBC_NOPAD       = 0x01,
CDRMC_ALG_SM4_CTR             = 0x02,
CDRMC_ ALG_AES_128_CBC_NOPAD  =0x03,
CDRMC_ALG_AES_128_CTR         = 0x04,
```

} CDRMC_Symmetric_Crypto_Algorithm;

描述：对称加解密算法及模式。

成员：

CDRMC_ALG_SM4_ CBC_NOPAD　　：SM4_CBC模式；

CDRMC_ALG_SM4 _CTR：SM4_CTR模式。

CDRMC_ ALG_AES_128_CBC_NOPAD：AES_128_CBC_NOPAD模式；

CDRMC_ALG_AES_128_CTR：AES_128_CTR模式。

C.1.3 CENC工作模式CDRMC_Cenc_ Algorithm

原型：typedef enum {

```
CDRMC_ALG_CENC_AES_CTR  = 0x01,
CDRMC_ALG_CENC_AES_CBC  = 0x02,
CDRMC_ALG_CENC_SM4_CTR  = 0x03,
CDRMC_ALG_CENC_SM4_CBC  = 0x04,
```

} CDRMC_Cenc_ Algorithm;
描述：CENC工作模式。
成员：
CDRMC_ALG_CENC_AES_CTR：AES-CTR模式；
CDRMC_ALG_CENC_AES_CBC：AES-CBC模式；
CDRMC_ALG_CENC_SM4_CTR：SM4-CTR模式；
CDRMC_ALG_CENC_SM4_CBC：SM4-CBC模式。

C.2 数据结构定义

C.2.1 Cipher句柄结构体CDRMC_SessionHandle

原型：typedef void* CDRMC_SessionHandle;
描述：DRM Session句柄结构定义。
成员：无。

C.2.2 CENC subsample结构体CDRMC_SubSample

```
原型：typedef struct __CDRMC_SubSample
{
    unsigned int u32ClearHeaderLen;
unsigned int u32PayLoadLen;
} CDRMC_SubSample;
```

描述：CENC subsample结构定义。
成员：
u32ClearHeaderLen：一个subsample中清流数据长度；
u32PayLoadLen：一个subsample中加密流对应的数据长度。

C.2.3 CENC数据结构体CDRMC_Cenc

```
原型：typedef struct __CDRMC_Cenc
{
    unsigned int u32KeyIdLen;
    unsigned char* pu8KeyId;
    unsigned int u32IVLen;
    unsigned char * pu8IV;
    unsigned int u32FirstEncryptOffset;
    CDRMC_SubSample * pstSubSample;
    unsigned int u32SubsampleNum;
} CDRMC_Cenc;
```

描述：CENC结构定义。
成员：
u32KeyIdLen：密钥标识长度；
pu8KeyId：密钥标识；
u32IVLen：初始向量长度；

pu8IV：初始向量；
u32FirstEncryptOffset：偏移地址；
pstSubSample：Subsample描述；
u32SubsampleNum：Subsample的数目。

C.3 接口定义

C.3.1 CDRMC_OpenSession接口

原型：int CDRMC_OpenSession (CDRMC_SessionHandle *phSession);
功能：创建DRM会话。
参数：phSession—输出参数，DRM会话句柄。
返回：int，0表示成功，其他表示失败。

C.3.2 CDRMC_CloseSession接口

原型：int CDRMC_CloseSession (CDRMC_SessionHandle hSession);
功能：关闭DRM会话。
参数：hSession—输入参数，DRM会话句柄。
返回：int，0表示成功，其他表示失败。

C.3.3 CDRMC_GetLicenseRequest接口

原型：int CDRMC_GetLicenseRequest (CDRMC_SessionHandle hSession, unsigned char* pu8DrmInfo, unsigned int u32DrmInfoLen, unsigned char* pu8LicenseRequest, unsigned int* pu32LicenseRequestLen);
功能：获取许可证请求消息。
参数：hSession—输入参数，DRM会话句柄；
pu8DrmInfo—输入参数，DRM信息；
u32DrmInfoLen—输入参数，DRM信息长度；
pu8LicenseRequest—输出参数，许可证请求数据缓冲区；
pu32LicenseRequestLen—输入输出参数，输入许可证请求数据缓冲区长度，输出许可证请求数据长度。
返回：int，0表示成功，其他表示失败。

C.3.4 CDRMC_ProcessLicenseResponse接口

原型：int CDRMC_ProcessLicenseResponse (CDRMC_SessionHandle hSession, unsigned char* pu8LicenseResponse, unsigned int u32LicenseResponseLen);
功能：处理许可证响应消息。
参数：hSession—输入参数，DRM会话句柄；
pu8LicenseResponse—输入参数，许可证数据；
u32LicenseResponseLen—输入参数，许可证数据长度。
返回：int，0表示成功，其他表示失败。

C.3.5 CDRMC_GetProvisionRequest接口

原 型： int CDRMC_GetProvisionRequest (CDRMC_SessionHandle hSession, unsigned char* pu8ProvisionRequest, unsigned int* pu32ProvisionRequestLen);

功能：获取证书申请消息。

参数：hSession—输入参数，DRM会话句柄；

pu8ProvisionRequest—输出参数，证书请求数据缓冲区；

pu32ProvisionRequestLen—输入输出参数，输入证书请求数据缓冲区长度，输出证书请求数据长度。

返回：int，0表示成功，其他表示失败。

C.3.6 CDRMC_ ProcessProvisionResponse接口

原型：int CDRMC_ProcessProvisionResponse (CDRMC_SessionHandle hSession, unsigned char* pu8ProvisionResponse, unsigned int u32ProvisionResponseLen);

功能：处理证书响应消息。

参数：hSession—输入参数，DRM会话句柄；

pu8ProvisionResponse—输入参数，证书数据；

u32ProvisionResponseLen—输入参数，证书数据长度。

返回：int，0表示成功，其他表示失败。

C.3.7 CDRMC_CheckRightsStatus接口

原 型： int CDRMC_CheckRightsStatus (CDRMC_SessionHandle hSession, unsigned char* pu8DrmInfo, unsigned int u32DrmInfoLen, CDRMC_Rights_Status *pRightsStatus);

功能：查询授权状态。

参数：hSession—输入参数，DRM会话句柄；

pu8DrmInfo—输入参数，DRM信息；

u32DrmInfoLen—输入参数，DRM信息长度；

pRightsStatus—输出参数，授权状态。

返回：int，0表示成功，其他表示失败。

C.3.8 CDRMC_Decrypt接口

原 型： int CDRMC_Decrypt (CDRMC_SessionHandle hSession, unsigned char* pu8DrmInfo, unsigned int u32DrmInfoLen, CDRMC_Symmetric_Crypto_Algorithm algo, unsigned char*pu8Iv, unsigned int u32IvLen, unsigned char*pu8Input, unsigned int u32InLen, unsigned char *pu8Output, unsigned int *pu32OutLen);

功能：解密。

参数：hSession—输入参数，DRM会话句柄；

pu8DrmInfo—输入参数，DRM信息；

u32DrmInfoLen—输入参数，DRM信息长度；

algo—输入参数，算法类型；

pu8Iv—输入参数，初始向量；

u32IvLen—输入参数，初始向量长度；

pu8Input—输入参数，需解密的数据；

u32InLen—输入参数，需解密数据的长度；

pu8Output—输出参数，解密后数据存储的缓冲区；

pu32OutLen—输入输出参数，输入解密后数据存储的缓冲区长度，输出解密后数据长度。

返回：int，0表示成功，其他表示失败。

C.3.9 CDRMC_Cenc_Decrypt

原型：int CDRMC_Cenc_Decrypt (CDRMC_SessionHandle hSession, unsigned char* pu8DrmInfo, unsigned int u32DrmInfoLen, CDRMC_Cenc_ Algorithm algo, CDRMC_Cenc* pstCENC, unsigned char*pu8Input, unsigned int u32InLen, unsigned char *pu8Output, unsigned int* pu32OutLen);

功能：CENC解密。

参数：hSession—输入参数，DRM会话句柄；

pu8DrmInfo—输入参数，DRM信息；

u32DrmInfoLen—输入参数，DRM信息长度；

algo—输入参数，算法类型；

pstCENC—输入参数，CENC结构体；

pu8Input—输入参数，需解密的数据；

u32InLen—输入参数，需解密数据的长度；

pu8Output—输出参数，解密后数据存储的缓冲区；

pu32OutLen—输入输出参数，输入解密后数据存储的缓冲区长度，输出解密后数据长度。

返回：int，0表示成功，其他表示失败。

C.3.10 CDRMC_GetDeviceId接口

原型：int CDRMC_GetDeviceId (CDRMC_SessionHandle hSession, unsigned char * pu8DeviceId, unsigned int* pu32DeviceIdLen);

功能：获取设备唯一标识。

参数：hSession—输入参数，DRM会话句柄；

pu8DeviceId—输出参数，设备唯一标识；

pu32DeviceIdLen—输入输出参数，输入设备唯一标识缓冲区长度，输出设备唯一标识长度。

返回int，0表示成功，其他表示失败。

C.3.11 CDRMC_GetDeviceCert接口

原型：int CDRMC_GetDeviceCert (CDRMC_SessionHandle hSession, unsigned char * pu8DeviceCert, unsigned int* pu32DeviceCertLen);

功能：获取设备证书。

参数：hSession—输入参数，DRM会话句柄；

pu8DeviceCert—输出参数，设备证书；

pu32DeviceCertLen—输入输出参数，输入设备证书缓冲区长度，输出设备证书长度。...

返回：int，0表示成功，其他表示失败。

C.3.12 CDRMC_GetDrmId接口

原型:int CDRMC_GetDrmId (CDRMC_SessionHandle hSession, unsigned char * pu8DrmId, unsigned int* pu32DrmIdLen);

功能：获取DRM厂商标识。

参数：hSession—输入参数，DRM会话句柄；

pu8DrmId—输出参数，DRM客户端厂商标识；

pu32DrmIdLen—输入输出参数，输入DRM客户端厂商标识缓冲区长度，输出DRM客户端厂商标识长度。

返回：int，0表示成功，其他表示失败。

C.4 接口调用流程

DRM功能模块接口调用流程如图C.1所示。

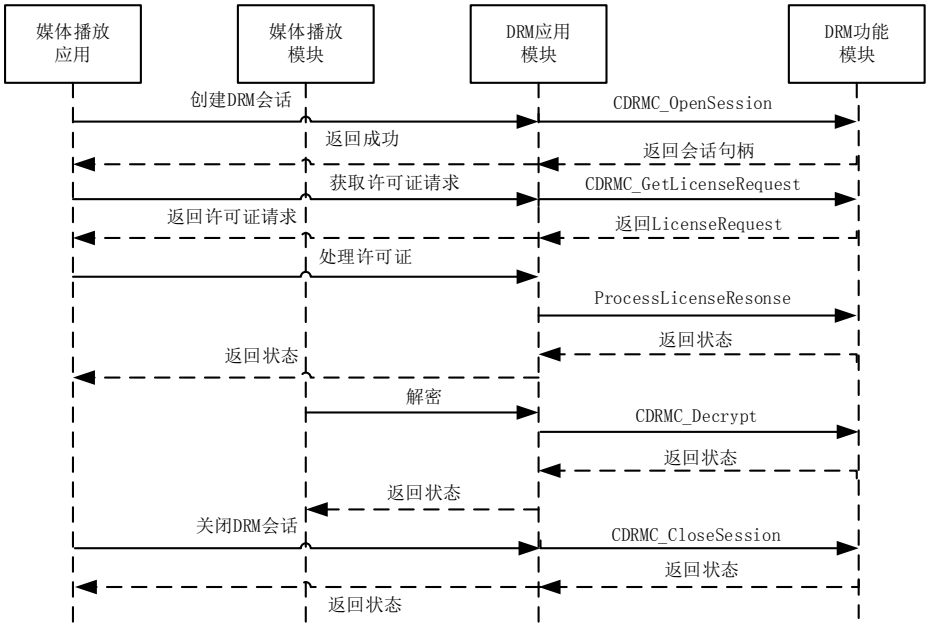

图 C.1 DRM 客户端功能模块接口调用流程

附 录 D
（规范性附录）
DRM 客户端运行环境接口

D.1 常量定义

D.1.1 RSA签名验签模式CDRMR_RSA_Sign_Algorithm

```
原型：typedef enum {
            CDRMR_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1  = 0x01,
            CDRMR_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256= 0x02,
            CDRMR_ALG_RSASSA_PKCS1_V1_5_SHA1      = 0x03,
            CDRMR_ALG_RSASSA_PKCS1_V1_5_SHA256     =0x04,
} CDRMR_RSA_Sign_Algorithm;
```

描述：RSA签名验签模式。

成员：

CDRMR_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1：RSASSA_PKCS1_PSS_MGF1_SHA1模式；

CDRMR_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256：RSASSA_PKCS1_PSS_MGF1_SHA256模式；

CDRMR_ALG_RSASSA_PKCS1_V1_5_SHA1：RSASSA_PKCS1_V1_5_SHA1模式；

CDRMR_ALG_RSASSA_PKCS1_V1_5_SHA256：RSASSA_PKCS1_V1_5_SHA256模式。

D.1.2 RSA加解密模式CDRMR_RSA_Crypto_Algorithm

```
原型：typedef enum {
            CDRMR_ALG_RSA_NOPAD                    = 0x01
            CDRMR_ALG_RSAES_PKCS1_V1_5             = 0x02,
            CDRMR_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1   = 0x03,
            CDRMR_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 = 0x04,
} CDRMR_RSA_Crypto_Algorithm;
```

描述：RSA加解密模式。

成员：

CDRMR_ALG_RSA_NOPAD：CDRMR_ALG_RSA_NOPAD模式；

CDRMR_ALG_RSAES_PKCS1_V1_5：RSAES_PKCS1_V1_5模式；

CDRMR_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1：CDRMR_RSAES_PKCS1_OAEP_MGF1_SHA1模式；

CDRMR_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256：RSAES_PKCS1_OAEP_MGF1_SHA256模式。

D.1.3 对称加解密算法及模式CDRMR_Symmetric_Crypto_Algorithm

```
原型：typedef enum {
                CDRMR_ALG_SM4_CBC_NOPAD     = 0x01,
                CDRMR_ALG_SM4_CTR           = 0x02,
                CDRMR_ALG_AES_128_CBC_NOPAD = 0x03,
                CDRMR_ALG_AES_128_CTR       = 0x04,
```

```
                    CDRMR_ALG_AES_256_CBC_NOPAD = 0x05,
                    CDRMR_ALG_AES_256_CTR       = 0x06,
                    CDRMR_ALG_SM4_ECB_NOPAD     = 0x07,
                    CDRMR_ALG_AES_128_ECB_NOPAD = 0x08,
                    CDRMR_ALG_AES_256_ECB_NOPAD = 0x09,
                    CDRMR_ALG_AES_WRAP          = 0x0a,
                    CDRMR_ALG_AES_UNWRAP        = 0x0b,
} CDRMR_Symmetric_Crypto_Algorithm;
```

描述：对称加解密算法及模式。

成员：

CDRMR_ALG_SM4_CBC_NOPAD：SM4_CBC_NOPAD模式；

CDRMR_ALG_SM4_CTR：SM4_CTR模式；

CDRMR_ALG_AES_128_CBC_NOPAD：AES_128_CBC_NOPAD模式；

CDRMR_ALG_AES_128_CTR：AES_128_CTR模式；

CDRMR_ALG_AES_256_CBC_NOPAD：AES_256_CBC_NOPAD模式；

CDRMR_ALG_AES_256_CTR：AES_256_CTR模式；

CDRMR_ALG_SM4_ECB_NOPAD　　：SM4_ECB模式；

CDRMR_ALG_AES_128_ECB_NOPAD：AES_128_ECB_NOPAD模式

CDRMR_ALG_AES_256_ECB_NOPAD：AES_256_ECB_NOPAD模式；

CDRMR_ALG_AES_WRAP：AES_WRAP模式；

CDRMR_ALG_AES_UNWRAP：AES_UNWRAP模式。

D.1.4　HASH算法及模式CDRMR_HASH_Algorithm

```
原型：typedef enum {
                        CDRMR_ALG_SM3   = 0x01,
                        CDRMR_ALG_SHA1  = 0x02,
                        CDRMR_ALG_SHA256= 0x03,
} CDRMR_HASH_Algorithm;
```

描述：HASH算法及模式。

成员：

CDRMR_ALG_SM3：SM3算法；

CDRMR_ALG_SHA1：SHA1算法；

CDRMR_ALG_SHA256：SHA256算法。

D.1.5　HMAC算法及模式CDRMR_HMAC_Algorithm

```
原型：typedef enum {
                      CDRMR_ALG_HMAC_SM3   = 0x01,
                      CDRMR_ALG_HMAC_SHA1  = 0x02,
                      CDRMR_ALG_HMAC_SHA256= 0x03,
} CDRMR_HMAC_Algorithm;
```

描述：HMAC算法及模式。

成员：

CDRMR_ALG_HMAC_SM3：SM3算法；

CDRMR_ALG_HMAC_SHA1：SHA1算法；
CDRMR_ALG_HMAC_SHA256：SHA256算法。

D.1.6 CENC工作模式CDRMR_Cenc_Algorithm

```
原型：typedef enum {
                    CDRMR_ALG_CENC_AES_CTR  = 0x01,
                    CDRMR_ALG_CENC_AES_CBC  = 0x02,
                    CDRMR_ALG_CENC_SM4_CTR  = 0x03,
                    CDRMR_ALG_CENC_SM4_CBC  = 0x04,
} CDRMR_Cenc_Algorithm;
```

描述：CENC工作模式。
成员：
CDRMR_ALG_CENC_AES_CTR：AES-CTR模式；
CDRMR_ALG_CENC_AES_CBC：AES-CBC模式；
CDRMR_ALG_CENC_SM4_CTR：SM4-CTR模式；
CDRMR_ALG_CENC_SM4_CBC：SM4-CBC模式。

D.1.7 存储类型CDRMR_Storage_Type

```
原型：typedef enum {
                      CDRMR_STORAGE_PRIVATE= 0x01,
                      CDRMR_STORAGE_RESERVED=0x02,
} CDRMR_Storage_Type;
```

描述：存储类型。
成员：
CDRMR_STORAGE_PRIVATE：私有类型；
CDRMR_STORAGE_RESERVED：保留。

D.1.8 存储权限标志CDRMR_Storage_Data_Access_Flag

```
原型：typedef enum {
              CDRMR_STORAGE_DATA_ACCESS_READ       = 0x01,
              CDRMR_STORAGE_DATA_ACCESS_WRITE      = 0x02,
              CDRMR_STORAGE_DATA_ACCESS_REMOVE     = 0x04,
              CDRMR_STORAGE_DATA_ACCESS_OVERWRITE  = 0x08,
              CDRMR_STORAGE_DATA_ACCESS_SHARE_READ = 0x10,
              CDRMR_STORAGE_DATA_ACCESS_SHARE_WRITE= 0x20,
} CDRMR_Storage_Data_Access_Flag;
```

描述：存储权限。
成员：
CDRMR_STORAGE_DATA_ACCESS_READ：读取对象的权限；
CDRMR_STORAGE_DATA_ACCESS_WRITE：写入对象的权限；
CDRMR_STORAGE_DATA_ACCESS_REMOVE：删除对象的权限；
CDRMR_STORAGE_DATA_ACCESS_OVERWRITE：更新对象的权限；
CDRMR_STORAGE_DATA_ACCESS_SHARE_READ：允许其他对象句柄读；

CDRMR_STORAGE_DATA_ACCESS_SHARE_WRITE：允许其他对象句柄写。

D.2 数据结构定义

D.2.1 Cipher句柄结构体CDRMR_CipherHandle

原型：typedef void* CDRMR_CipherHandle;
描述：Cipher句柄结构定义。
成员：无。

D.2.2 Hash句柄结构体CDRMR_HashHandle

原型：typedef void* CDRMR_HashHandle;
描述：Hash句柄结构定义。
成员：无。

D.2.3 HMAC句柄结构体CDRMR_HMacHandle

原型：typedef void* CDRMR_HMacHandle;
描述：HMAC句柄结构定义。
成员：无。

D.2.4 安全存储对象句柄结构体CDRMR_SecureStorageObjectHandle

原型：typedef void* CDRMR_SecureStorageObjectHandle;
描述：安全存储对象句柄结构定义。
成员：无。

D.2.5 CENC subsample结构体CDRMR_SubSample

原型：typedef struct __CDRMR_SubSample
{
    unsigned int u32ClearHeaderLen;
    unsigned int u32PayLoadLen;
} CDRMR_SubSample;
描述：CENC subsample结构定义。
成员：
u32ClearHeaderLen：一个subsample中清流数据长度；
u32PayLoadLen：一个subsample中加密流对应的数据长度。

D.2.6 CENC数据结构体CDRMR_Cenc

原型：typedef struct __CDRMR_Cenc
{
   unsigned int u32KeyLen;
   unsigned char* pu8Key;
   unsigned int u32IVLen;
   unsigned char * pu8IV;

```
    unsigned int u32FirstEncryptOffset;
      CDRMR_SubSample_S * pstSubSample;
      unsigned int u32SubsampleNum;
} CDRMR_Cenc;
```

描述：CENC结构定义。

成员：

u32KeyLen：密钥长度；

pu8Key：密钥；

u32IVLen：初始向量长度；

pu8IV：初始向量；

u32FirstEncryptOffset：偏移地址；

pstSubSample：Subsample描述；

u32SubsampleNum：Subsample的数目。

D.2.7 时间结构体CDRMR_Time

```
原型：typedef struct __ CDRMR_Time {
          unsigned int seconds;
          unsigned int millis;
 } CDRMR_Time;
```

描述：时间结构定义，从1970年1月1日0时开始的时间。

成员：

seconds：秒。

millis：毫秒。

D.3 接口定义

D.3.1 密码算法

D.3.1.1 CDRMR_Crypto_Sm2Sign接口

原型：int CDRMR_Crypto_Sm2Sign(ECCrefPrivateKey *pstECCPrivateKey, unsigned char* pu8InData, unsigned int u32InDataLen, ECCSignature *pstECCSignature);

功能：SM2私钥签名。

参数：pstECCPrivateKey—输入参数，SM2私钥，ECCrefPrivateKey见GB/T 36322—2018；

pu8InData—输入参数，需签名数据；

u32InDataLen—输入参数，需签名数据长度；

pstECCSignature—输出参数，签名数据，ECCSignature见GB/T 36322—2018。

返回：int，0表示成功，其他表示失败。

D.3.1.2 CDRMR_Crypto_Sm2Verify接口

原型：int CDRMR_Crypto_Sm2Verify(ECCrefPublicKey *pstECCPublicKey, unsigned char* pu8InData, unsigned int u32InDataLen, ECCSignature *pstECCSignature);

功能：SM2公钥验签。

参数：pstECCPublicKey—输入参数，SM2公钥，ECCrefPublicKey见GB/T 36322—2018；
pu8InData—输入参数，需签名数据；
u32InDataLen—输入参数，需签名数据长度；
pstECCSignature—输入参数，签名数据，ECCSignature见GB/T 36322—2018。

返回：int，0表示成功，其他表示失败。

D.3.1.3 CDRMR_Crypto_Sm2Encrypt接口

原型：int CDRMR_Crypto_Sm2Encrypt(ECCrefPublicKey *pstECCPublicKey, unsigned char*pu8Input, unsigned int u32InLen, ECCCipher *pstECCCipher);

功能：SM2公钥加密。

参数：pstECCPublicKey—输入参数，SM2公钥，ECCrefPublicKey见GB/T 36322—2018；
pu8Input—输入参数，需加密数据；
u32InLen—输入参数，需加密数据长度；
pstECCCipher—输出参数，加密的数据，ECCCipher见GB/T 36322—2018。

返回：int，0表示成功，其他表示失败。

D.3.1.4 CDRMR_Crypto_Sm2Decrypt接口

原型：int CDRMR_Crypto_Sm2Decrypt(ECCrefPrivateKey *pstECCPrivateKey, unsigned char *pu8Output, unsigned int* pu32OutLen, ECCCipher *pstECCCipher);

功能：SM2私钥解密。

参数：pstECCPrivateKey—输入参数，SM2私钥，ECCrefPrivateKey见GB/T 36322—2018；
pu8Output—输出参数，解密后数据；
u32OutLen—输入输出参数，输入解密后数据的缓冲区长度，输出解密后数据长度；
pstECCCipher—输入参数，加密的数据， ECCCipher见GB/T 36322—2018。

返回：int，0表示成功，其他表示失败。

D.3.1.5 CDRMR_Crypto_RsaSign接口

原型：int CDRMR_Crypto_RsaSign(CDRMR_RSA_Sign_Algorithm algo, RSArefPrivateKey *pstRSAPrivateKey, unsigned char *pu8InData, unsigned int u32InDataLen, unsigned char *pu8HashData, unsigned int u32HashDataLen, unsigned char *pu8OutSign, unsigned int *pu32OutSignLen);

功能：RSA私钥签名。

参数：algo—输入参数，签名算法模式；
pstRSAPrivateKey—输入参数，RSA私钥，RSArefPrivateKey见GB/T 36322—2018；
pu8InData—输入参数，需签名数据；
u32InDataLen—输入参数，需签名数据长度；
pu8HashData—输入参数，需签名的摘要数据；
u32HashDataLen—输入参数，需签名的摘要数据长度；
pu8OutSign—输出参数，签名数据；
pu32OutSignLen—输入输出参数，输入签名数据缓冲区长度，输出签名数据长度。

返回：int，0表示成功，其他表示失败。

D.3.1.6 CDRMR_Crypto_RsaVerify接口

原型：int CDRMR_Crypto_RsaVerify(CDRMR_RSA_Sign_Algorithm algo, RSArefPublicKey *pstRSAPublicKey, unsigned char *pu8InData, unsigned int u32InDataLen, unsigned char *pu8HashData, unsigned int u32HashDataLen, unsigned char *pu8InSign, unsigned int u32InSignLen);

功能：RSA公钥验签。

参数：algo—输入参数，验签算法模式；

pstRSAPublicKey—输入参数，RSA公钥，RSArefPublicKey见GB/T 36322—2018；

pu8InData—输入参数，需验证签名的数据；

u32InDataLen—输入参数，需验证签名的数据的长度；

pu8HashData—输入参数，需验证签名的摘要数据；

u32HashDataLen—输入参数，需验证签名的摘要数据的长度；

pu8InSign—输入参数，签名数据；u32InSignLen—输入参数，签名数据长度。

返回：int，0表示成功，其他表示失败。

D.3.1.7 CDRMR_Crypto_RsaEncrypt接口

原型：int CDRMR_Crypto_RsaEncrypt(CDRMR_RSA_Crypto_Algorithm algo, RSArefPublicKey *pstRSAPublicKey, unsigned char*pu8Input, unsigned int u32InLen, unsigned char *pu8Output, unsigned int *pu32OutLen);

功能：RSA公钥加密。

参数：algo—输入参数，加密算法模式；

pstRSAPublicKey—输入参数，RSA公钥，RSArefPublicKey见GB/T 36322—2018；

pu8Input—输入参数，需加密数据；

u32InLen—输入参数，需加密数据长度；

pu8Output—输出参数，存储加密后的数据的缓冲区；

pu32OutLen—输入输出参数，输入存储加密后的数据的缓冲区的长度，输出加密后的数据的长度。

返回：int，0表示成功，其他表示失败。

D.3.1.8 CDRMR_Crypto_RsaDecrypt接口

原型：int CDRMR_Crypto_RsaDecrypt(CDRMR_RSA_Crypto_Algorithm algo, RSArefPrivateKey *pstRSAPrivateKey, unsigned char*pu8Input, unsigned int u32InLen, unsigned char *pu8Output, unsigned int *pu32OutLen);

功能：RSA私钥解密。

参数：algo—输入参数，解密算法模式；

pstRSAPrivateKey—输入参数，RSA私钥，RSArefPrivateKey见GB/T 36322—2018；

pu8Input—输入参数，需解密数据；

u32InLen—输入参数，需解密数据长度；

pu8Output—输出参数，存储解密后的数据的缓冲区；

pu32OutLen—输入输出参数，输入存储解密后的数据的缓冲区的长度，输出解密后的数据的长度。

返回：int，0表示成功，其他表示失败。

D.3.1.9 CDRMR_Crypto_SymmetricEncrypt接口

原型：int CDRMR_Crypto_SymmetricEncrypt (CDRMR_Symmetric_Crypto_Algorithm algo, unsigned char * pu8Key, unsigned int u32KeyLen, unsigned char* pu8Iv, unsigned int u32IvLen, unsigned char*pu8Input, unsigned int u32InLen, unsigned char *pu8Output, unsigned int *pu32OutLen);

功能：对称算法加密。

参数：algo—输入参数，加密算法及模式；

pu8Key—输入参数，加密钥；

u32KeyLen—输入参数，加密密钥长度；

pu8Iv—输入参数，加密初始向量；

u32IvLen—输入参数，加密初始向量长度；

pu8Input—输入参数，需加密数据；

u32InLen—输入参数，需加密数据长度；

pu8Output—输出参数，存储加密后的数据的缓冲区；

pu32OutLen—输入输出参数，输入存储加密后的数据的缓冲区的长度，输出加密后的数据的长度。

返回：int，0表示成功，其他表示失败。

D.3.1.10 CDRMR_Crypto_SymmetricDecrypt接口

原型：int CDRMR_Crypto_SymmetricDecrypt (CDRMR_Symmetric_Crypto_Algorithm algo, unsigned char * pu8Key, unsigned int u32KeyLen, unsigned char* pu8Iv, unsigned int u32IvLen, unsigned char*pu8Input, unsigned int u32InLen, unsigned char *pu8Output, unsigned int *pu32OutLen);

功能：对称算法解密。

参数：algo—输入参数，加密算法及模式；

pu8Key—输入参数，解密密钥；

u32KeyLen—输入参数，解密密钥长度；

pu8Iv—输入参数，解密初始向量；

u32IvLen—输入参数，解密初始向量长度；

pu8Input—输入参数，需解密数据；

u32InLen—输入参数，需解密数据长度；

pu8Output—输出参数，存储解密后的数据的缓冲区；

pu32OutLen—输入输出参数，输入存储解密后的数据的缓冲区的长度，输出解密后的数据的长度。

返回：int，0表示成功，其他表示失败。

D.3.1.11 CDRMR_Crypto_HashInit接口

原型：int CDRMR_Crypto_HashInit (CDRMR_HASH_Algorithm algo, CDRMR_HashHandle *phHashHandle);

功能：初始化Hash算法句柄。

参数：algo—输入参数，摘要算法；

phHashHandle—输出参数，Hash句柄。

返回：int，0表示成功，其他表示失败。

D.3.1.12 CDRMR_Crypto_HashUpdate接口

原 型： int CDRMR_Crypto_HashUpdate (CDRMR_HashHandle hHashHandle, unsigned char *pu8InputData, unsigned int u32InputDataLen);

功能： 更新需要计算Hash的数据。

参数：hHashHandle—输入参数，Hash句柄；

pu8InputData—输入参数，需计算Hash的数据；

u32InputDataLen—输入参数，需计算Hash的数据长度

返回：int，0表示成功，其他表示失败。

D.3.1.13 CDRMR_Crypto_HashDoFinal接口

原 型： int CDRMR_Crypto_HashDoFinal (CDRMR_HashHandle hHashHandle, unsigned char *pu8OutputHash, unsigned int* pu32OutputHashLen);

功能：计算Hash。

参数：hHashHandle—输入参数，Hash句柄；

pu8OutputHash—输出参数，存储Hash值的缓冲区；

pu32OutputHashLen—输入输出参数，输入存储Hash值的缓冲区大小，输出实际Hash值长度。

返回：int，0表示成功，其他表示失败。

D.3.1.14 CDRMR_Crypto_HmacInit接口

原型：int CDRMR_Crypto_HmacInit (CDRMR_HMAC_Algorithm algo, unsigned char *pu8Key, unsigned int pu32KeyLen, CDRMR_HMacHandle *phHmacHandle);

功能：初始化HMAC算法句柄。

参数：algo—输入参数，HMAC算法；

pu8Key—输入参数，密钥；

pu32KeyLen—输入参数，密钥长度；

phHmacHandle—输出参数，HMAC句柄。

返回：int，0表示成功，其他表示失败。

D.3.1.15 CDRMR_Crypto_HmacUpdate接口

原 型： int CDRMR_Crypto_HmacUpdate (CDRMR_HMacHandle hHmacHandle, unsigned char *pu8InputData, unsigned int u32InputDataLen);

功能： 更新需要计算HMAC的数据。

参数：hHmacHandle—输入参数，HMAC句柄；

pu8InputData—输入参数，需计算HMAC的数据；

u32InputDataLen—输入参数，需计算HMAC的数据长度。

返回：int，0表示成功，其他表示失败。

D.3.1.16 CDRMR_Crypto_HmacDoFinal接口

原 型： int CDRMR_Crypto_HmacDoFinal (CDRMR_HMacHandle hHmacHandle, unsigned char *pu8OutputHmac, unsigned int* pu32OutputHmacLen);

功能：计算HMAC。

参数：hHmacHandle—输入参数，HMAC句柄；

pu8OutputHmac—输出参数，存储HMAC值的缓冲区；

pu32OutputHmacLen—输入输出参数，输入存储HMAC值的缓冲区大小，输出实际HMAC值长度。

返回：int，0表示成功，其他表示失败。

D.3.1.17 CDRMR_Cipher_Init接口

原型：int CDRMR_Cipher_Init(void);
功能：初始化cipher设备。
参数：无。
返回：int，0表示成功，其他表示失败。

D.3.1.18 CDRMR_Cipher_DeInit接口

原型：int CDRMR_Cipher_DeInit(void);
功能：去初始化cipher设备。
参数：无。
返回：int，0表示成功，其他表示失败。

D.3.1.19 CDRMR_Cipher_CreateHandle接口

原型：int CDRMR_Cipher_CreateHandle(CDRMR_CipherHandle *phCipher, void* pCipherReserved);
功能：获取cipher句柄。
参数：phCipher—输出参数，Cipher句柄；
pCipherReserved—输入参数，保留扩展参数。
返回：int，0表示成功，其他表示失败。

D.3.1.20 CDRMR_Cipher_DestroyHandle接口

原型：int CDRMR_Cipher_DestroyHandle(CDRMR_CipherHandle hCipher);
功能：释放cipher句柄。
参数：hCipher—输入参数，cipher句柄；
返回：int，0表示成功，其他表示失败。

D.3.1.21 CDRMR_Cipher_ConfigHandle接口

原型：int CDRMR_Cipher_ConfigHandle(CDRMR_CipherHandle hCipher, CDRMR_Symmetric_Crypto_Algorithm algo, unsigned char *pu8Key, unsigned int pu32KeyLen, unsigned char *pu8Iv, unsigned int pu32IvLen);
功能：配置cipher控制参数。
参数：hCipher—输入参数，cipher句柄；
algo—输入参数，加解密算法及模式；
pu8Key—输入参数，密钥；
pu32KeyLen—输入参数，密钥长度；
pu8Iv—输入参数，初始化向量；
pu32IvLen—输入参数，初始化向量长度。
返回：int，0表示成功，其他表示失败。

D.3.1.22 CDRMR_Cipher_Copy接口

原型：int CDRMR_Cipher_Copy(CDRMR_CipherHandle hCipher, unsigned int u32NonSecInputPhyAddr, unsigned int u32SecOutputPhyAddr, unsigned int u32ByteLength);

功能：完成非安全物理地址到安全物理地址的内存拷贝。

参数：hCipher—输入参数，cipher句柄；

u32NonSecInputPhyAddr—输入参数，非安全侧物理地址；

u32SecOutputPhyAddr—输入参数，安全侧物理地址；

u32ByteLength—输入参数，要拷贝的数据长度。

返回：int，0表示成功，其他表示失败。

D.3.1.23 CDRMR_Cipher_Encrypt接口

原型：int CDRMR_Cipher_Encrypt(CDRMR_CipherHandle hCipher, unsigned int u32SrcPhyAddr, unsigned int u32DestPhyAddr, unsigned int u32ByteLength);

功能：加密。

参数：hCipher—输入参数，cipher句柄；

u32SrcPhyAddr—输入参数，要加密的数据存放的物理地址；

u32DestPhyAddr—输入参数，加密后的数据存放的物理地址；

u32ByteLength—输入参数，要加密的数据长度。

返回：int，0表示成功，其他表示失败。

D.3.1.24 CDRMR_Cipher_Decrypt接口

原型：int CDRMR_Cipher_Decrypt(CDRMR_CipherHandle hCipher, unsigned int u32SrcPhyAddr, unsigned int u32DestPhyAddr, unsigned int u32ByteLength);

功能：解密。

参数：hCipher—输入参数，cipher句柄；

u32SrcPhyAddr—输入参数，要解密的数据存放的物理地址；

u32DestPhyAddr—输入参数，解密后的数据存放的物理地址；

u32ByteLength—输入参数，要解密的数据长度。

返回：int，0表示成功，其他表示失败。

D.3.1.25 CDRMR_Cipher_CENCDecrypt接口

原型：int CDRMR_Cipher_CENCDecrypt(CDRMR_CipherHandle hCipher, CDRMR_Cenc_Algorithm algo, CDRMR_Cenc* pstCENC, unsigned char* pu8InputPhyAddr, unsigned int u32InputLen, unsigned char* pu8OutputPhyAddr, unsigned int* pu32OutputLen);

功能：CENC解密。

参数：hCipher—输入参数，cipher句柄；

algo—输入参数，CENC工作模式；

pstCENC—输入参数，CENC结构体；

pu8InputPhyAddr—输入参数，需解密数据的物理地址；

u32InputLen—输入参数，需解密数据长度；

pu8OutputPhyAddr—输入参数，用于存储解密后的数据的缓冲区的物理地址；

pu32OutputLen—输入输出参数，输入解密后数据存储缓冲区的长度，输出解密后的数据的长度。

返回：int，0表示成功，其他表示失败。

D.3.2 安全存储

D.3.2.1 CDRMR_SecureStorage_CreateObject接口

原型：int CDRMR_SecureStorage_CreateObject(CDRMR_Storage_Type storageType, CDRMR_Storage_Data_Access_Flag flags, unsigned char *pu8ObjectId, unsigned int u32ObjectIdLen, CDRMR_SecureStorageObjectHandle* phObjectHandle);

功能：创建安全存储对象。

参数：storageType—输入参数，安全存储类型；

flags—输入参数，创建安全存储对象的权限；

pu8ObjectId—输入参数，安全存储对象标识；

u32ObjectIdLen—输入参数，安全存储对象标识长度；

phObjectHandle—输出参数，安全存储对象句柄。

返回：int，0表示成功，其他表示失败。

D.3.2.2 CDRMR_SecureStorage_OpenObject接口

原型：int CDRMR_SecureStorage_OpenObject(CDRMR_Storage_Type storageType, CDRMR_Storage_Data_Access_Flag flags, unsigned char *pu8ObjectId, unsigned int u32ObjectIdLen, CDRMR_SecureStorageObjectHandle* phObjectHandle);

功能：打开安全存储对象。

参数：storageType—输入参数，安全存储类型；

flags—输入参数，访问安全存储对象的权限；

pu8ObjectId—输入参数，安全存储对象标识；

u32ObjectIdLen—输入参数，安全存储对象标识长度；

phObjectHandle—输出参数，安全存储对象句柄。

返回：int，0表示成功，其他表示失败。

D.3.2.3 CDRMR_SecureStorage_ReadObjectData接口

原型：int CDRMR_SecureStorage_ReadObjectData(CDRMR_SecureStorageObjectHandle hObjectHandle, unsigned char* pu8Buffer, unsigned int* pu32BufferLen);

功能：读取安全存储对象数据。

参数：hObjectHandle—输入参数，安全存储对象句柄；

pu8Buffer—输出参数，安全存储对象数据缓冲区；

pu32BufferLen—输入输出参数，输入安全存储对象数据缓冲区长度，返回安全存储对象数据长度。

返回：int，0表示成功，其他表示失败。

D.3.2.4 CDRMR_SecureStorage_WriteObjectData接口

原型：int CDRMR_SecureStorage_WriteObjectData(CDRMR_SecureStorageObjectHandle hObjectHandle, unsigned char* pu8Buffer, unsigned int u32BufferLen);

功能：写入安全存储对象数据。

参数：hObjectHandle—输入参数，安全存储对象句柄；

pu8Buffer—输入参数，需写入安全存储对象的数据；

u32BufferLen—输入参数，写入安全存储对象数据长度。

返回：int，0表示成功，其他表示失败。

D.3.2.5 CDRMR_SecureStorage_GetObjectSize接口

原型：int CDRMR_SecureStorage_GetObjectSize (CDRMR_SecureStorageObjectHandle hObjectHandle, unsigned int* pu32ObjectDataLen);

功能：读取安全存储对象数据长度。

参数：hObjectHandle—输入参数，安全存储对象句柄；

pu32ObjectDataLen—输出参数，安全存储对象数据长度。

返回：int，0表示成功，其他表示失败。

D.3.2.6 CDRMR_SecureStorage_CloseObject接口

原型：int CDRMR_SecureStorage_CloseObject(CDRMR_SecureStorageObjectHandle hObjectHandle);

功能：关闭安全存储对象。

参数：hObjectHandle—输入参数，安全存储对象句柄。

返回：int，0表示成功，其他表示失败。

D.3.2.7 CDRMR_SecureStorage_CloseAndRemoveObject接口

原型：int CDRMR_SecureStorage_CloseAndRemoveObject(CDRMR_SecureStorageObjectHandle hObjectHandle);

功能：关闭并移除安全存储对象。

参数：hObjectHandle—输入参数，安全存储对象句柄。

返回：int，0表示成功，其他表示失败。

D.3.2.8 CDRMR_SecureStorage_WriteOTP接口

原型：int CDRMR_SecureStorage_WriteOTP(unsigned char* pu8Data, unsigned int u32DataLen, unsigned int u32OTPAddr);

功能：将数据写入OTP。

参数：pu8Data—输入参数，需写入的数据；

u32DataLen—输入参数，需写入的数据的长度；

u32OTPAddr—输入参数，写入OTP的地址。

返回：int，0表示成功，其他表示失败。

D.3.2.9 CDRMR_SecureStorage_LockOTP接口

原型：int CDRMR_SecureStorage_LockOTP(unsigned int u32OTPLockAddr);

功能：锁住OTP地址。

参数：u32OTPLockAddr—输入参数，OTP Lock地址。

返回：int，0表示成功，其他表示失败。

D.3.2.10 CDRMR_SecureStorage_ReadOTP接口

原型：int CDRMR_SecureStorage_ReadOTP(unsigned char* pu8OutputData, unsigned int u32OutputDataLen, unsigned int u32OTPAddr);

功能：读取OTP中数据。

参数：pu8OutputData—输出参数，数据缓冲区；
u32OutputDataLen—输入参数，需要读取的数据长度；
u32OTPAddr—输入参数，OTP地址。
返回：int，0表示成功，其他表示失败。

D.3.3 随机数CDRMR_Random_GetNumber接口

原型：int CDRMR_Random_GetNumber(unsigned int *pu32RandomNumber);
功能：获取随机数。
参数：pu32RandomNumber—输出参数，返回随机数。
返回：int，0表示成功，其他表示失败。

D.3.4 安全内存

D.3.4.1 CDRMR_SecureMemory_Malloc接口

原型：void* CDRMR_SecureMemory_Malloc (unsigned int u32Size);
功能：分配内存。
参数：u32Size—输入参数，分配内存的长度。
返回：void*，非NULL表示成功，NULL表示失败。

D.3.4.2 CDRMR_SecureMemory_Free接口

原型：void CDRMR_SecureMemory_Free(void* buffer);
功能：释放内存。
参数：buffer—输入参数，缓冲区地址。
返回：无。

D.3.4.3 CDRMR_SecureMemory_Memcpy接口

原型：void* CDRMR_SecureMemory_Memcpy (void* dest, void* src, unsigned int u32Size);
功能：内存拷贝。
参数：dest—输出参数，目标缓冲区地址；
src—输入参数，源缓冲区地址；
u32Size—输入参数，数据长度。
返回：void*， 返回指向dest的指针，NULL表示失败。

D.3.4.4 CDRMR_SecureMemory_Memcmp接口

原型：int CDRMR_SecureMemory_Memcmp(void* buffer1, void* buffer2, unsigned int u32Size);
功能：内存比较。
参数：buffer1—输入参数，缓冲区地址；
Buffer2—输入参数，缓冲区地址；
u32Size—输入参数，数据长度。
返回：0表示buffer1和buffer2相等，大于0表示buffer1大于buffer2，小于0表示buffer1小于buffer2。

D.3.4.5 CDRMR_SecureMemory_Memset接口

原型：void* CDRMR_SecureMemory_Memset(void* buffer, int s32Value, unsigned int u32Size);
功能：将buffer中当前位置后面的n个字节设置为s32Value后返回buffer。
参数：buffer—输出参数，缓冲区地址；
s32Value—输入参数，设置的数值；
u32Size—输入参数，数据长度。
返回：void*，buffer指针表示成功，NULL表示失败。

D.3.5 安全时间

D.3.5.1 CDRMR_Time_GetSystemTime接口

原型：void CDRMR_Time_GetSystemTime(CDRMR_Time *time);
功能：获取系统时间。
参数：time—输出参数，系统时间。
返回：无。

D.3.5.2 CDRMR_Time_GetTAPersistentTime接口

原型：int CDRMR_Time_GetTAPersistentTime(CDRMR_Time *time);
功能：获取为可信应用TA维护的时间，需使用CDRMR_Time_SetTAPersistentTime函数设置后才能调用成功。
参数：time—输出参数，时间。
返回：0表示成功，其他表示失败。

D.3.5.3 CDRMR_Time_SetTAPersistentTime接口

原型：int CDRMR_Time_SetTAPersistentTime(CDRMR_Time *time);
功能：为可信应用TA设置可维护的时间。
参数：time—输入参数，时间。
返回：0表示成功，其他表示失败。

D.3.5.4 CDRMR_Time_GetREETime接口

原型：void CDRMR_Time_GetREETime(CDRMR_Time *time);
功能：获取REE时间。
参数：time—输出参数，REE时间。
返回：无。

D.3.6 输出控制

D.3.6.1 CDRMR_OutputControl_GetMaxCapability接口

原型：int CDRMR_OutputControl_GetMaxCapability (unsigned int u32Type, void* caps);
功能：获取支持的最大输出控制能力。
参数：u32Type—输入参数，输出控制能力类型，支持数字输出控制、模拟输出控制、数字水印等类型；
caps—输出参数，输出控制能力描述。
返回：0表示成功，其他表示失败。

D.3.6.2 CDRMR_OutputControl_GetCurrentCapabilityStatus接口

原型：int CDRMR_OutputControl_GetCurrentCapabilityStatus (unsigned int u32Type, void *status);

功能：获取当前设置的输出控制能力。

参数：u32Type—输入参数，输出控制能力类型，支持数字输出控制、模拟输出控制、数字水印等类型；

status—输出参数，当前输出控制能力状态。

返回：0表示成功，其他表示失败。

D.3.6.3 CDRMR_OutputControl_ConfigCapability接口

原型：int CDRMR_OutputControl_ConfigCapability (unsigned int u32Type, void *params);

功能：设置输出控制能力参数。

参数：u32Type—输入参数，输出控制能力类型，支持数字输出控制、模拟输出控制、数字水印等类型；

params—输入参数，输出控制能力参数。

返回：0表示成功，其他表示失败。

中　华　人　民　共　和　国
广播电视行业标准
视音频内容分发数字版权管理技术规范
GY/T 277－2019
*
国家广播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.abp2003.cn
北京复兴门外大街二号
联系电话：（010）86093424　　86092923
邮政编码：100866