# 流媒体概念

针对哪类碎片化协议？HLS？OSMF？

流媒体（streaming media）是指将媒体数据压缩后，通过互联网分段的发送数据，达到在互联网上即时传输媒体数据以供观赏的一种技术。此技术使得媒体数据能够像流水一样发送，因此称之为流媒体技术。与传统媒体相比，流媒体技术有以下几个特点：

1. 用户不需要下载完整的媒体文件就能播放媒体文件，还能实时的更改媒体文件的播放位置，用户播放媒体文件后本地也不会留下媒体文件。
2. 内容提供方不仅能提供已经存在的媒体文件，还能提供实时产生的媒体数据，比如摄像头实时采集的音视频数据、OBS软件实时采集的屏幕实况。
3. 内容提供方可以对正在传输的媒体文件片段进行压缩，降低对网络带宽的依赖。

目前，比较流行的流媒体协议有以下几种：

## 实时传输协议/实时传输控制协议（RTP/RTCP）

**A．实时传输协议（RTP）**

RTP（Real-time Transport Protocol，实时传输协议）是由IETF的多媒体传输工作小组提出的一个协议，对应的RFC文档为RFC3550（RFC1889为其过期版本）。RFC3550还定义了RTCP协议，RTP协议和RTCP协议会协同工作来提供流媒体服务。

RTP协议是介于应用层协议和传输层协议之间的协议，采用通过单播或多播的方式为语音、图像、传真等多种多媒体数据提供端到端的实时传输服务，**RTP的最典型应用场景是多人音视频会议。**

RTP协议在默认情况下采用UDP协议来传输媒体数据，因此RTP本身的服务质量是没有保证的，并且RTP协议本身缺乏协调同一RTP会话中不同RTP流的机制。所以RTP协议总是与RTCP（Real-time Transport Control Protocol，实时传输控制协议）协同工作来保证服务质量。

当应用程序启动一个RTP会话时将同时占用两个端口，分别供RTP和RTCP使用，通常RTCP会采用与RTP相同的分发机制，向会话中所有成员周期性地发送控制信息。RTCP收集相关RTP流的统计信息，例如：传输字节数，传输分组数，丢失分组数，时延抖动，单向和双向网络延迟等等。网络应用程序可以利用RTCP所提供的信息提高RTP协议的服务质量，比如限制信息流量或改用压缩比较小的编解码器。

应用程序通过接收这些数据，从中获取会话参与者的相关资料，以及网络状况、分组丢失概率等反馈信息。

RTP数据包头格式：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| V | P | X | CC | M | PT | Sqeuence Number |
| Timestamp | | | | | | |
| Synchronization Source (SSRC) identifier | | | | | | |
| Contributing Source (CSRC) identifier …… | | | | | | |
|

上面各个字段含义如下：

1. Version（V）：2 bits大小，RTP版本号，现在用的是2。
2. Padding（P）：1 bit大小，如果设置了该字段，报文的末尾会包含一个或多个填充字节，这些填充字节不是payload的内容。最后一个填充字节标识了总共需要忽略多少个填充字节（包括自己）。
3. Extension（X）：1 bit大小，如果设置了该字段，那么RTP报文头后有拓展数据。
4. CSRC count（CC）：4 bits大小，CSRC列表的长度。
5. Marker（M）：1 bit大小，Marker在报文流中用它来划分每一帧的边界。可以在RTP预设中定义附加的marker，或者移除Marker来拓展PT字段的长度。
6. Payload type（PT）: 7 bits大小，该字段定义RTP payload的格式和他在预设中的意义。上层应用可能会定义一个（静态的类型码<->payload格式）映射关系。也可以用RTP协议外的方式来动态地定义payload类型。
7. Sequence number：16 bits大小，每发送一个RTP包则该序列号加1，RTP包的接受者可以通过它来确定丢包情况并且利用它来重排包的顺序，这个字段的初始值应该是随机的。
8. Timestamp：32 bits大小，时间戳反映了RTP数据包生成第一块数据时的时刻。这个时间戳必须恒定地线性增长，因为它会被用来同步数据包和计算网络抖动，也就是说即使暂停发包小段时间，Timestamp也需要增长。
9. SSRC：32 bits大小，该字段用来确定数据的发送源。这个身份标识应该随机生成，并且要保证同一个RTP session中没有重复的SSRC。同一个SSRC中发送的所有包都具有同一时序和序列号间隔，一个信号源（麦克风，摄像头）的报文流会有由一个SSRC的发送器发送。一个SSRC可能会随着时间的变化，改变其数据格式，例如音频编码。如果一个与会者在一个RTP session中发送不同的媒体数据流，那么每个流的SSRC必须不同。
10. CSRC list：有0 ~ 15个条目，32 bits大小，each CSRC list表示对该payload数据做出贡献的所有SSRC。这个字段包含的SSRC数量由CC字段定义。如果有超过15个SSRC，只有15个可以被记录。

RTP默认是采用UDP发送的，因此以上格式为使用UDP作为底层协议的RTP头部格式。如果要使用TCP作为底层协议，那么需要在RTP头之前再加上四个字节。

1. 第一个字节：$，辨识符
2. 第二个字节：通道，在SETUP的过程中获取
3. 第三第四个字节： RTP包的大小，最多只能12位，第三个字节保存到高4位，第四个字节保存低8位

**B．实时传输控制协议（RTCP）**

RTCP控制协议需要与RTP数据协议一起配合使用，当应用程序启动一个RTP会话时将同时占用两个端口，分别供RTP和RTCP使用，通常RTCP会采用与RTP相同的分发机制，向会话中所有成员周期性地发送控制信息，应用程序通过接收这些数据，从中获取会话参与者的相关资料，以及网络状况、分组丢失概率等反馈信息。它的主要功能有如下几点：

1. 反馈数据分发的质量，进行流量控制和拥塞控制。
2. RTCP给每个RTP source的不同RTP流（音频、视频）带一个不变的传输层身份识别符（CNAME）。发送时间戳来同步不同RTP的时间。
3. 通过每个参与者都广播RTCP控制报文的方式，适当的控制RTP报文发送的频率以至于RTP协议可以在大量客户端一同加入时也能正常工作。
4. 提供一个RTP协议的多个终端能够简单通信的机制。

RTCP定义了四种报文结构来传输不同的控制信息，：

1. SR：发送端报告，所谓发送端是指发出RTP数据报的应用程序或者终端，发送端同时也可以是接收端。(SERVER定时间发送给CLIENT)。
2. RR：接收端报告，所谓接收端是指仅接收但不发送RTP数据报的应用程序或者终端。(SERVER接收CLIENT端发送过来的响应)。
3. SDES：源描述，主要功能是作为会话成员有关标识信息的载体，如用户名、邮件地址、电话号码等，此外还具有向会话成员传达会话控制信息的功能。
4. BYE：通知离开，主要功能是指示某一个或者几个源不再有效，即通知会话中的其他成员自己将退出会话。
5. APP：由应用程序自己定义，解决了RTCP的扩展性问题，并且为协议的实现者提供了很大的灵活性。

## 1.2 实时传输协议/实时传输控制协议（RTSP）

RTSP（Real Time Streaming Protocol，实时流协议）是由Real Network和Netscape共同提出的一种双向实时流媒体数据传输协议。它向客户端提供了诸如暂停、快进、回放、快进等控制，因此**RTSP协议最广泛的应用是视频点播服务。**

RTSP协议本身不传输媒体数据，而是控制底层的RTP/RTCP协议来传输数据，除此之外RTSP还可以选择TCP、UDP、组播UDP等通道来发送数据，具有很好的拓展性。RTSP的作用相当于流媒体服务器的远程控制。除了利用RTP/RTCP协议来传输数据之外，RTSP协议还可以选择TCP、UDP、组播UDP等通道来发送数据，具有很好的扩展性。RTSP协议并不特别强调时间同步，所以比较能容忍网络延迟。

1.2中提到过RTP/RTCP都属于介于传输层和应用层之间的协议，而RTSP协议是构建在RTP/RTCP协议之上的应用层协议。他们的关系图如下所示：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
|  | 应用层（RTSP） | | | | |  |
|
|  | | | | |
|  | | | | |
|  | RTP/RTCP |  | TCP |  |
| UDP |
| 传输层 | | | | |
|
|  | | | | |
| 网络层（IP） | | | | |
|
|  | | | | |
| 数据链路层 | | | | |
|
|  | | | | |
| 物理层 | | | | |
|
|  | | | | |

RTSP的交互过程就是客户端请求，服务器响应，并且RTSP的请求和响应的报文格式非常类似于HTTP协议请求、响应的报文格式，请求报文有多行请求行组成；响应行同样由多行响应头组成。

RTSP协议的请求格式为：

|  |
| --- |
| method url vesion\r\n  CSeq: x\r\n  xxx\r\n  ...  \r\n |

其中“method”参数表明请求的方法。“url”参数表明请求的URL，格式为“rtsp://ip:port/session”，ip表示服务端主机IP，port表示服务端端口号、如果不写那么就是默认端口 554，session表明请求哪一个会话。“version”参数表明RTSP协议的版本（现在的版本为RTSP/1.0）。 “CSeq”参数表明请求的序列号。RTSP定义了很多请求方法，常用的有以下几个：

|  |  |
| --- | --- |
| 方法 | 描述 |
| OPTIONS | 获得服务端提供的可用方法 |
| DESCRIBE | 向服务端获取对应会话的媒体描述信息 |
| SETUP | 向服务端发起建立请求，建立连接会话 |
| PLAY | 向服务端发起播放请求 |
| TEARDOWN | 向服务端发起关闭连接会话请求 |

RTSP协议的响应格式为：

|  |
| --- |
| vesion 200 OK\r\n  CSeq: x\r\n  xxx\r\n  ...  \r\n |

“version”参数表明RTSP协议的版本（现在的版本为RTSP/1.0）。“CSeq”参数表示响应的序列号，响应的序列号必须与请求的序列号相同。

## 1.3 Real Time Messaging Protocol（RTMP）

RTMP（Real Time Messaging Protocol）是Adobe Systems公司为Flash播放器和服务器之间音频、视频和数据传输开发的开放协议。RTMP协议是应用层协议，同时具有高度可扩展性。

播放RTMP协议的传输的流媒体数据需要Flash插件，现在几乎所有的Windows-PC都安装有Flash插件，再加上RTMP 是专为流媒体开发的协议，对底层的优化比其它协议更加优秀，所以RTMP协议的应用非常广泛。但是无论安卓设备还是苹果设备对没有在系统组件级别上支持Flash插件，其次Windows-PC上的Flash插件因为安全问题，正在被逐渐淘汰，比如“谷歌浏览器”已经宣布2020年12月以后不再支持Flash插件。

因此，**目前RTMP协议最主要的应用场景是“推流”，也就是媒体资源生产者通过RTMP协议把媒体数据推送给媒体资源分发者。**媒体资源分发者一般不会通过RTMP协议把媒体资源推送给媒体资源点播者。

RTMP协议涉及两个重要的概念：多路服用和分包。当RTMP协议在互联网中传输数据的时候，发送方首先把媒体数据封装成消息（Message），消息会被拆分成更小的单元，称为消息块（Chunk），最后将分割后的消息块通过TCP协议发送给客户端。客户端在通过TCP协议收到数据后，首先把消息块重新组合成消息，然后通过对消息进行解封装处理就可以恢复出媒体数据。

RTMP协议这样的工作模式使得RTMP协议能够在一条TCP连接中并行的传输多种不同的媒体资源。RTMP协议之上的应用层协议还可以为不同的媒体资源指定不同的优先级，这样在网络带宽受限时能优先传输重要的媒体资源。

## 1.4 HTTP Live Streaming（HLS）

HLS协议由苹果公司开发，旨在能够在苹果全系列产品中避免使用不安全的Flash插件。目前**HLS协议已经成为使用最广泛的流媒体播放协议之一**，在Apple的全系列产品iPhone、iPad、MAC和Android设备都在系统组件级别支持播放HLS。在Windows-PC端，通过JavaScript编写的HLS解码器，所有的版本高的浏览器也支持播放HLS。但是由于HLS延迟较高，比较少用在直播上。

HLS 的工作原理是将一个媒体资源分成一个个小的文件块（MPEG-TS格式），将所有分片媒体资源路径按时间次序记录于 m3u8 文件（即 playlist）内，其中附带一些额外描述（比如该资源的加密方式，每个分片的媒体播放时长）。

HLS的关键就是m3u8文件，m3u8文件可分为两类：媒体播放列表（Media Playlist）和主播放列表（Master Playlist）。

当 m3u8 文件作为主播放列表时，其内部提供的是同一份媒体资源的多份不同格式的媒体播放列表文件，比如说同一份媒体资源的不同码率，不同拍摄角度。客户可以根据不同的网络状态选取合适码率的资源，并且最好根据用户喜好选择合适的资源内容。

当 m3u8 文件作为媒体播放列表时，它承担实际播放媒体资源的任务，记录着媒体资源的实际URL、加密方式等内容

在开始一个流媒体会话时，客户端可以基于 HTTP 的GET请求先下载m3u8文件，然后根据m3u8文件中的内容使用HTTP 的GET请求下载媒体资源片段，然后按次序播放这些资源分片。如果服务端能够不断生成最新的媒体文件片段，HLS播放器就能不断的播放，达到直播的效果。

HLS 只请求基本的 HTTP 报文，与RTP/RTCP/RTSP/RMTP不同，HLS 可以穿过任何允许 HTTP 数据通过的防火墙或者代理服务器。它也很容易使用内容分发网络来传输媒体流。

m3u8 的文件格式主要包含三方面内容：

1. **文件播放列表格式定义**：播放列表（Playlist，也即 m3u8 文件）内容需严格满足规范定义所提要求。下面罗列一些主要遵循的条件：
2. m3u8文件必须utf-8进行编码，不能使用Byte Order Mark（BOM）字节序，不能包含utf-8控制字符（U+0000~U\_001F和U+007F~U+009F）。
3. m3u8文件的每一行要么是一个URI，要么是空行，要么就是以#开头的字符串。不能出现空白字符，除了显示声明的元素。
4. m3u8 文件中以#开头的字符串要么是注释，要么就是标签。标签以#EXT开头，大小写敏感。
5. **属性列表：**某些特定的标签的值为属性列表。标签后面的属性列表以逗号作为分隔符，分离出多组不带空格的“属性/值”对。

“属性/值”对的语法格式如下：

|  |
| --- |
| AttributeName=AttributeValue |

其中：属性名是由[A..Z],[0..9]和-组成的不带引号的字符串。并且属性名和“=”号中间不能有空格，“=”和属性值之间也不能有空格。

属性值只能取以下类型：

1. 十进制整型（decimal-interger）：由[0..9]之间组成的十进制不带引号的字符串，范围为0-264（18446744073709551615），字符长度为1-20之间。
2. 十六进制序列：由[0..9]和[A..F]且前缀为0x或0X组合成的不带引号的字符串。其序列的最大长度取决于他的属性名。
3. 带符号十进制浮点型（signed-decimal-floating-point）：由[0..9]，“-”和“.”组合成的不带引号的字符串。
4. 字符串（quoted-string）：由双引号包裹表示的字符串。其中，0xA，0xD 和双引号不能出现在该字符串中。该字符串区分大小写。
5. 可枚举字符串（enumerated-string）：由属性名显示定义的一系列不带引号的字符串。该字符串不能包含双引号，逗号和空白字符。
6. 十进制分辨率：由字符x进行隔离的两个十进制整型数。第一个整型表示视频水平宽度大小，第二个整型数表示视频垂直方向高度大小（单位：像素）。
7. **标签：**标签用于指定 m3u8 文件的全局参数或在其后面的切片文件/媒体播放列表的一些信息。标签的类型可分为五种类型：[基础标签（Basic Tags）](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23section-4.3.1)，[媒体片段类型标签（Media Segment Tags）](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-13)，[媒体播放列表类型标签](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-22)，[主播放列表类型标签](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-25)和[播放列表类型标签](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-35)。其具体内容如下所示：
8. [**基础标签（Basic Tags）**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23section-4.3.1)**：**可同时适用于媒体播放列表（Media Playlist）和主播放列表（Master Playlist）。具体标签如下：
   1. [EXTM3U](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-12)：表明该文件是一个m3u8文件。每个 [M3U](https://links.jianshu.com/go?to=https://zh.wikipedia.org/wiki/M3U) 文件必须将该标签放置在第一行。
   2. [EXT-X-VERSION](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-12)：表示 [HLS](https://links.jianshu.com/go?to=https://zh.wikipedia.org/wiki/HTTP_Live_Streaming) 的协议版本号，该标签与流媒体的兼容性相关。该标签为全局作用域，使能整个 m3u8 文件；每个 m3u8 文件内最多只能出现一个该标签定义。如果 m3u8 文件不包含该标签，则默认为协议的第一个版本。
9. [**媒体片段类型标签（Media Segment Tags）**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-13)**：**每个切片 URI 前面都有一系列媒体片段标签对其进行描述。有些片段标签只对其后切片资源有效；有些片段标签对其后所有切片都有效，直到后续遇到另一个该标签描述。媒体片段类型标签不能出现在主播放列表（Master Playlist）中。具体标签如下：
10. [**EXTINF**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-13)**：**表示其后 URL 指定的媒体片段时长（单位为秒）。每个 URL 媒体片段之前必须指定该标签。格式中duration可以为十进制的整型或者浮点型，其值必须小于或等于EXT-X-TARGETDURATION指定的值。该标签只对其后一个URI起作用。建议始终使用浮点型指定时长，这可以让客户端在定位流时，减少四舍五入错误。但是如果兼容版本号EXT-X-VERSION小于3，那么必须使用整型。格式：#EXTINF:<duration>,[<title>]
11. [**EXT-X-BYTERANGE**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-14)**：**该标签表示接下来的切片资源是其后URI指定的媒体片段资源的局部范围（即截取URI媒体资源部分内容作为下一个切片）。该标签只对其后一个URI起作用。格式中：n是一个十进制整型，表示截取片段大小（单位：字节）。可选参数o也是一个十进制整型。格式：#EXT-X-BYTERANGE:<n>[@<o>]
12. [**EXT-X-DISCONTINUITY**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-14)**：**该标签表明其前一个切片与下一个切片之间存在中断。当以下任一种情况变化时，必须使用该标签：文件格式（file format），数字（number），类型（type），媒体标识符（identifiers of tracks），时间戳序列（timestamp sequence）。当以下任一情况变化时，应当使用该标签：编码参数（encoding parameters），编码序列（encoding sequence）。EXT-X-DISCONTINUITY 的一个经典使用场景就是在视频流中插入广告，由于资源视频流与广告视频流不是同一份资源，因此在这两种流切换时使用 EXT-X-DISCONTINUITY 进行指明。格式：#EXT-X-DISCONTINUITY

1. **[EXT-X-KEY](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-15" \t "_blank)：**媒体片段可以进行加密，而该标签可以指定解密方法。该标签对所有媒体片段和由标签#EXT-X-MAP声明的围绕其间的所有媒体初始化块（Meida Initialization Section）都起作用，直到遇到下一个 EXT-X-KEY（若 m3u8 文件只有一个#EXT-X-KEY标签，则其作用于所有媒体片段）。多个EXT-X-KEY标签如果最终生成的是同样的密钥，则他们都可作用于同一个媒体片段。该标签的属性列表可以包含如下几个键。**METHOD：**该值是一个可枚举的字符串，指定了加密方法。该键是必须参数。其值可为NONE，AES-128，SAMPLE-AES当中的一个。其中NONE表示切片未进行加密（此时其他属性不能出现）；AES-128：表示表示使用 [AES-128](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23ref-AES_128) 进行加密。**URI：**指定密钥路径。该密钥是一个 16 字节的数据。该键是必须参数，除非METHOD为NONE。SAMPLE-AES：意味着媒体片段当中包含样本媒体，比如音频或视频，它们使用[AES-128](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23ref-AES_128)进行加密。这种情况下IV属性可以出现也可以不出现。**IV：**该值是一个 128 位的十六进制数值。[AES-128](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23ref-AES_128)要求使用相同的 16字节IV值进行加密和解密。使用不同的IV值可以增强密码强度。如果属性列表出现IV，则使用该值；如果未出现，则默认使用媒体片段序列号（即EXT-X-MEDIA-SEQUENCE）作为其IV值，使用大端字节序，往左填充0直到序列号满足16字节（128位）。**KEYFORMAT：**标识密钥在密钥文件中的存储方式。非必须参数，默认是“identity”，密钥文件中的AES-128密钥是以二进制方式存储的16个字节的密钥。**KEYFORMATVERSIONS：**可选参数，KEYFORMAT有多个版本时存在。由“/”分隔的字符串（如“1/3”），如果同一KEYFORMAT有多个版本，则该属性存在，用来区分KEYFORMAT的不同版本。格式：#EXT-X-KEY:<attribute-list>。
2. **KEYFORMAT：**由双引号包裹的字符串，标识密钥在密钥文件中的存储方式（密钥文件中的 [AES-128](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23ref-AES_128) 密钥是以二进制方式存储的16个字节的密钥）。该属性为可选参数，其默认值为"identity"。使用该属性要求兼容版本号EXT-X-VERSION大于等于5。
3. **KEYFORMATVERSIONS：**由一个或多个被/分割的正整型数值构成的带引号的字符串（比如："1"，"1/2"，"1/2/5"）。如果有一个或多特定的KEYFORMT版本被定义了，则可使用该属性指示具体版本进行编译。该属性为可选参数，其默认值为"1"。使用该属性要求兼容版本号EXT-X-VERSION大于等于5。
4. [**EXT-X-MAP**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-17)**：**该标签指明了获取媒体初始化块（Meida Initialization Section）的方法。该标签对其后所有媒体片段生效，直至遇到另一个 EXT-X-MAP 标签。该标签的属性列表可以包含如下几个键。**URI：**由引号包裹的字符串，指定了包含媒体初始化块的资源的路径。该属性为必选参数。**BYTERANGE：**由引号包裹的字符串，指定了媒体初始化块在 URI 指定的资源的位置（片段）。该属性指定的范围应当只包含媒体初始化块。该属性为可选参数，如果未指定，则表示 URI 指定的资源就是全部的媒体初始化块。其格式为：#EXT-X-MAP:<attribute-list>。
5. [**EXT-X-PROGRAM-DATE-TIME**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-18)**：**该标签使用一个绝对日期/时间表明第一个样本片段的取样时间。其格式为：#EXT-X-PROGRAM-DATE-TIME:<date-time-msec>。其中，date-time-msec是一个 ISO/IEC 8601:2004 规定的日期格式，形如：YYYY-MM-DDThh:mm:ss.SSSZ。
6. **EXT-X-DATERANGE：**该标签定义了一系列由属性/值对组成的日期范围。其格式为：#EXT-X-DATERANGE:<attribute-list>。该标签的属性列表可以包含如下几个键。**ID：**双引号包裹的唯一指明日期范围的标识。该属性为必选参数。**CLASS：**双引号包裹的由客户定义的一系列属性与与之对应的语意值。所有拥有同一 CLASS 属性的日期范围必须遵守对应的语意。该属性为可选参数。**START-DATE：**双引号包裹的日期范围起始值。该属性为必选参数。**END-DATE：**双引号包裹的日期范围结束值。该属性值必须大于或等于 START-DATE。该属性为可选参数。**DURATION：**日期范围的持续时间是一个十进制浮点型数值类型（单位：秒）。该属性值不能为负数。**PLANNED-DURATION：**该属性为日期范围的期望持续时长。其值为一个十进制浮点数值类型（单位：秒）。该属性值不能为负数。在预先无法得知真实持续时长的情况下，可使用该属性作为日期范围的期望预估时长。该属性为可选参数。**X-<client-attribute>：**X-前缀是预留给客户端自定义属性的命名空间。客户端自定义属性名时，应当使用反向 DNS（reverse-DNS）语法来避免冲突。自定义属性值必须是使用双引号包裹的字符串，或者是十六进制序列，或者是十进制浮点数，比如：X-COM-EXAMPLE-AD-ID="XYZ123"。该属性为可选参数。**SCTE35-CMD, SCTE35-OUT, SCTE35-IN：**用于携带 SCET-35 数据。该属性为可选参数。**END-ON-NEXT：**该属性值为一个可枚举字符串，其值必须为YES。该属性表明达到该范围末尾，也即等于后续范围的起始位置 START-DATE。后续范围是指具有相同 CLASS 的，在该标签 START-DATE 之后的具有最早 START-DATE 值的日期范围。该属性时可选参数。
7. [**媒体播放列表类型标签**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-22)**：**媒体播放列表标签为 m3u8 文件的全局参数信息。这些标签只能在 m3u8 文件中至多出现一次。媒体播放列表（Media Playlist）标签不能出现在主播放列表（Master Playlist）中。媒体播放列表具体标签如下所示：
8. [**EXT-X-TARGETDURATION**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-22)**：**表示每个视频分段最大的时长（单位秒）。该标签为必选标签。其格式为：#EXT-X-TARGETDURATION:<s>其中：参数s表示目标时长（单位：秒）。
9. [**EXT-X-MEDIA-SEQUENCE**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-22)**：**表示播放列表第一个 URL 片段文件的序列号。每个媒体片段 URL 都拥有一个唯一的整型序列号。每个媒体片段序列号按出现顺序依次加1。如果该标签未指定，则默认序列号从0开始。媒体片段序列号与片段文件名无关。其格式为：#EXT-X-MEDIA-SEQUENCE:<number>。其中：参数number即为切片序列号。
10. [**EXT-X-DISCONTINUITY-SEQUENCE**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-23)**：**该标签使能同步相同流的不同 Rendition 和具备EXT-X-DISCONTINUITY标签的不同备份流。其格式为：#EXT-X-DISCONTINUITY-SEQUENCE:<number>。其中：参数number为一个十进制整型数值。如果播放列表未设置 EXT-X-DISCONTINUITY-SEQUENCE 标签，那么对于第一个切片的中断序列号应当为 0。
11. [**EXT-X-ENDLIST**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-23)**：**表明m3u8文件的结束。该标签可出现在m3u8文件任意位置，一般是结尾。其格式为：#EXT-X-ENDLIST
12. [**EXT-X-PLAYLIST-TYPE**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-24)**：**表明流媒体类型。全局生效。该标签为可选标签。其格式为：#EXT-X-PLAYLIST-TYPE:<type-enum>。其中type-enum可选值如下。VOD：即 Video on Demand，表示该视频流为点播源，因此服务器不能更改该m3u8文件；EVENT：表示该视频流为直播源，因此服务器不能更改或删除该文件任意部分内容（但是可以在文件末尾添加新内容）。注：VOD文件通常带有EXT-X-ENDLIST标签，因为其为点播源，不会改变；而EVEVT文件初始化时一般不会有EXT-X-ENDLIST标签，暗示有新的文件会添加到播放列表末尾，因此也需要客户端定时获取该m3u8文件，以获取新的媒体片段资源，直到访问到EXT-X-ENDLIST标签才停止）。
13. [**EXT-X-I-FRAMES-ONLY**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-24)**：**该标签表示每个媒体片段都是一个I-frame。I-frames帧视频编码不依赖于其他帧数，因此可以通过I-frame 进行快速播放，急速翻转等操作。该标签全局生效。其格式为：#EXT-X-I-FRAMES-ONLY。如果播放列表设置了EXT-X-I-FRAMES-ONLY，那么切片的时长（EXTINF标签的值）即为当前切片I-frame帧开始到下一个I-frame帧出现的时长。媒体资源如果包含I-frame切片，那么必须提供媒体初始化块或者通过EXT-X-MAP标签提供媒体初始化块的获取途径，这样客户端就能通过这些I-frame切片以任意顺序进行加载和解码。如果I-frame切片设置了EXT-BYTERANGE，那么就绝对不能提供媒体初始化块。使用EXT-X-I-FRAMES-ONLY要求的兼容版本号EXT-X-VERSION大于等于4。
14. [**主播放列表类型标签**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-25)**：**主播放列表（Master Playlist）定义了备份流，多语言翻译流和其他全局参数。主播放列表标签绝不能出现在媒体播放列表（Media Playlist）中。其具体标签如下：
15. [**EXT-X-MEDIA**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-25)**：**用于指定相同内容的可替换的多语言翻译播放媒体列表资源。比如，通过三个EXT-X-MEIDA标签，可以提供包含英文，法语和西班牙语版本的相同内容的音频资源，或者通过两个EXT-X-MEDIA提供两个不同拍摄角度的视频资源。其格式为：#EXT-X-MEDIA:<attribute-list>。该标签的属性列表可以包含如下几个键。**TYPE：**该属性值为一个可枚举字符串。其值有如下四种：AUDIO，VIDEO，SUBTITLES，CLOSED-CAPTIONS。通常使用的都是CLOSED-CAPTIONS。该属性为必选参数。**URI：**双引号包裹的媒体资源播放列表路径。如果TYPE属性值为CLOSED-CAPTIONS，那么则不能提供URI。该属性为可选参数。**GROUP-ID：**双引号包裹的字符串，表示多语言翻译流所属组。该属性为必选参数。**LANGUAGE：**双引号包裹的字符串，用于指定流主要使用的语言。该属性为可选参数。**ASSOC-LANGUAGE：**双引号包裹的字符串，其内包含一个语言标签，用于提供多语言流的其中一种语言版本。该参数为可选参数。**NAME：**双引号包裹的字符串，用于为翻译流提供可读的描述信息。如果设置了LANGUAGE属性，那么也应当设置NAME属性。该属性为必选参数。**DEFAULT：**该属性值为一个可枚举字符串。可选值为YES和NO。该属性未指定时默认值为NO。如果该属性设为YES，那么客户端在缺乏其他可选信息时应当播放该翻译流。该属性为可选参数。**AUTOSELECT：**该属性值为一个可枚举字符串。其有效值为YES或NO。未指定时，默认设为NO。如果该属性设置YES，那么客户端在用户没有显示进行设置时，可以选择播放该翻译流，因为其能配置当前播放环境，比如系统语言选择。如果设置了该属性，那么当DEFAULT设置YES时，该属性也必须设置为YES。该属性为可选参数。**FORCED：**该属性值为一个可枚举字符串。其有效值为YES或NO。未指定时，默认设为NO。只有在设置了TYPE为SUBTITLES时，才可以设置该属性。当该属性设为YES时，则暗示该翻译流包含重要内容。当设置了该属性，客户端应当选择播放匹配当前播放环境最佳的翻译流。当该属性设为NO时，则表示该翻译流内容意图用于回复用户显示进行请求。该属性为可选参数。**INSTREAM-ID：**由双引号包裹的字符串，用于指示切片的语言（Rendition）版本。当 TYPE 设为CLOSED-CAPTIONS时，必须设置该属性。其可选值为："CC1", "CC2", "CC3", "CC4" 和"SERVICEn"（n的值为 1~63）。对于其他TYPE值，该属性绝不能进行设置。**CHARACTERISTICS：**由双引号包裹的由一个或多个由逗号分隔的URI构成的字符串。每个URI表示一种翻译流的特征。该属性可包含私有URI。该属性为可选参数。**CHANNELS：**由双引号包裹的有序，由反斜杠/分隔的参数列表组成的字符串。所有音频 EXT-X-MEDIA 标签应当都设置CHANNELS属性。如果主播放列表包含两个相同编码但是具有不同数目channed的翻译流，则必须设置CHANNELS属性；否则，CHANNELS属性为可选参数。
16. [**EXT-X-STREAM-INF**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-29)**：**该属性指定了一个备份源。该属性值提供了该备份源的相关信息。其格式为：#EXT-X-STREAM-INF:<attribute-list><URI>。其中：URI 指定的媒体播放列表携带了该标签指定的翻译备份源。URI 为必选参数。该标签的属性列表可以包含如下几个键。**BANDWIDTH：**该属性为每秒传输的比特数，也即带宽。代表该备份流的巅峰速率。该属性为必选参数。**AVERAGE-BANDWIDTH：**该属性为备份流的平均切片传输速率。该属性为可选参数。**CODECS：**双引号包裹的包含由逗号分隔的格式列表组成的字符串。每个EXT-X-STREAM-INF标签都应当携带 CODECS 属性。**RESOLUTION：**该属性描述备份流视频源的最佳像素方案。该属性为可选参数，但对于包含视频源的备份流建议增加该属性设置。**FRAME-RATE：**该属性用一个十进制浮点型数值作为描述备份流所有视频最大帧率。对于备份流中任意视频源帧数超过每秒 30 帧的，应当增加该属性设置。该属性为可选参数，但对于包含视频源的备份流建议增加该属性设置。**HDCP-LEVEL：**该属性值为一个可枚举字符串。其有效值为TYPE-0或NONE。值为TYPE-0表示该备份流可能会播放失败，除非输出被高带宽数字内容保护（HDCP）。值为NONE表示流内容无需输出拷贝保护。使用不同程度的 HDCP 加密备份流应当使用不同的媒体加密密钥。该属性为可选参数。在缺乏 HDCP 可能存在播放失败的情况下，应当提供该属性。**AUDIO：**属性值由双引号包裹，其值必须与定义在主播放列表某处的设置了 TYPE 属性值为AUDIO的EXT-X-MEDIA 标签的 GROUP-ID 属性值相匹配。该属性为可选参数。**VIDEO：**属性值由双引号包裹，其值必须与定义在主播放列表某处的设置了TYPE属性值为VIDEO 的EXT-X-MEDIA标签的GROUP-ID属性值相匹配。该属性为可选参数。**SUBTITLES：**属性值由双引号包裹，其值必须与定义在主播放列表某处的设置了TYPE属性值为SUBTITLES的EXT-X-MEDIA 标签的GROUP-ID属性值相匹配。该属性为可选参数。**CLOSED-CAPTIONS：**该属性值可以是一个双引号包裹的字符串或NONE。如果其值为一个字符串，则必须与定义在主播放列表某处的设置了TYPE属性值为CLOSED-CAPTIONS的EXT-X-MEDI标签的GROUP-ID属性值相匹配。如果其值为NONE，则所有的ext-x-stream-inf标签必须同样将该属性设置NONE，表示主播放列表备份流均没有关闭的标题。对于某个备份流具备关闭标题，另一个备份流不具备关闭标题可能会触发播放中断。该属性为可选参数。
17. [**EXT-X-I-FRAME-STREAM-INF**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-33)**：**该标签表明媒体播放列表文件包含多种媒体资源的I-frame帧。其格式为：#EXT-X-I-FRAME-STREAM-INF:<attribute-list>。该标签的属性列表包含了EXT-X-I-FRAME-STREAM-INF标签同样的属性列表选项，除了FRAME-RATE，AUDIO，SUBTITLES和CLOSED-CAPTIONS。除此之外，其他的属性还有：URI：该属性值由双引号包裹的字符串，指示了I-frame 媒体播放列表文件的路径，该媒体播放列表文件必须包含EXT-X-I-FRAMES-ONLY标签。
18. [**EXT-X-SESSION-DATA**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-34)**：**该标签允许主播放列表携带任意 session 数据。该标签为可选参数。其格式为：#EXT-X-SESSION-DATA:<attribute-list>。该标签的属性列表可以包含如下几个键。**DATA-ID：**由双引号包裹的字符串，代表一个特定的数据值。该属性应当使用反向 DNS 进行命名，如"com.example.movie.title"。然而，由于没有中央注册机构，所以可能出现冲突情况。该属性为必选参数。**VALUE：**该属性值为一个双引号包裹的字符串，其包含 DATA-ID 指定的值。如果设置了 LANGUAGE，则 VALUE 应当包含一个用该语言书写的可读字符串。**URI：**由双引号包裹的 URI 字符串。由该 URI 指示的资源必选使用 JSON 格式，否则，客户端可能会解析失败。**LANGUAGE：**由双引号包裹的，包含一个语言标签的字符串。指示了 VALUE 所使用的语言。
19. [**EXT-X-SESSION-KEY**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-35)**：**该标签允许主播放列表（Master Playlist）指定媒体播放列表（Meida Playlist）的加密密钥。这使得客户端可以预先加载这些密钥，而无需从媒体播放列表中获取。该标签为可选参数。其格式为：#EXT-X-SESSION-KEY:<attribute-list>。其属性列表与EXT-X-KEY相同，除了METHOD属性的值必须不为NONE。
20. [**播放列表类型标签**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-35)**：**以下标签可同时设置于主播放列表（Master Playlist）和媒体播放列表（Media Playlist）中。但是对于在主播放列表中设置了的标签，不应当再次设置在主播放列表指向的媒体播放列表中。同时出现在两者播放列表的相同标签必须具备相同的值。这些标签在播放列表中不能出现多次（只能使用一次）。具体标签如下所示：
21. [**EXT-X-INDEPENDENT-SEGMENTS**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-35)**：**该标签表明对于一个媒体片段中的所有媒体样本均可独立进行解码，而无须依赖其他媒体片段信息。该标签对列表内所有媒体片段均有效。其格式为：#EXT-X-INDEPENDENT-SEGMENTS。如果该标签出现在主播放列表中，则其对所有媒体播放列表的所有媒体片段都生效。
22. [**EXT-X-START**](https://links.jianshu.com/go?to=https://tools.ietf.org/html/rfc8216%23page-36)**：**该标签表示播放列表播放起始位置。默认情况下，客户端开启一个播放会话时，应当使用该标签指定的位置进行播放。该标签为可选标签。其格式为：#EXT-X-START:<attribute-list>。
23. **TIME-OFFSET：**该属性值为一个带符号十进制浮点数（单位：秒）。一个正数表示以播放列表起始位置开始的时间偏移量。一个负数表示播放列表上一个媒体片段最后位置往前的时间偏移量。该属性的绝对值应当不超过播放列表的时长。如果超过，则表示到达文件结尾（数值为正数），或者达到文件起始（数值为负数）。如果播放列表不包含 EXT-X-ENDLIST 标签，那么TIME-OFFSET属性值不应当在播放文件末尾三个切片时长之内。

## 1.5 HTTP-FLV

FLV (Flash Video) 是Adobe公司推出的另一种视频格式，是一种在网络上传输的流媒体数据存储容器格式。其格式相对简单轻量，不需要很大的媒体头部信息。整个FLV由FLV Header, FLV Body组成，FLV Body又由Tag和每个Tag前面的PreviousTagSize组成。FLV Tag分为脚本Tag，也就是记录了媒体资源的格式，视频Tag，音频Tag。采用 FLV 格式封装的文件后缀为.flv。

而HTTP-FLV协议就是将流媒体数据封装成FLV格式，然后通过HTTP 协议传输给客户端。HTTP-FLV依靠MIME的特性，根据协议中的Content-Type来选择相应的程序去处理相应的内容，使得流媒体可以通过HTTP传输。而且HTTP首部行中有content-length字段，记录了http的body部分的长度。服务器回复http请求的时候如果有这个字段，客户端就接收这个长度的数据然后就认为数据传输完成了，如果服务器回复http请求中没有这个字段，客户端就一直接收数据，直到服务器跟客户端的socket连接断开。这样就是实现FLV格式文件的流式传输的原理。

# 流媒体加密趋势

付费点播流媒体的模式是很多平台的核心业务，如果媒体被盗取并非法传播，付费业务将受到严重威胁。因此对流媒体服务进行加密的技术变得尤为重要。

流媒体加密是为了让要保护的媒体资源不能被下载，即使下载到了也是加密后的内容，其它人解开加密后的内容需要付出非常大的代价。

加密整个视频的技术很简单，把因为视屏本身就是一个问价，按照文件加密即可。但对能够对流媒体传输的内容加密的流媒体就不多了。HLS协议原生的支持对流媒体加密，RTMP协议本身并不支持流媒体加密，但是变种版本RTMPS、RTMPE提供了流媒体加密功能，

# 流媒体加密的原理

以HLS的为例，根据1.4中HLS协议中m3u8文件的结构，可知HLS流媒体加密功能主要是由媒体片段类型标签中的#EXT-X-KEY标签实现的。

#EXT-X-KEY的格式为：#EXT-X-KEY:<attribute-list>。#EXT-X-KEY的属性值有5个，在1.4小节中已经详细的讲述过了。HLS流媒体加密的实现方式是对TS切片文件直接加密。

HLS中媒体分块如果是加密的，其加密密钥通过M3U8文件中的#EXT-X-KEY来指定，密钥文件由客户端从服务器请求认证获得。一个播放列表可以有一个以上的#EXT-X-KEY，同一个媒体段也可以有多个不同KEYFORMAT属性值的#EXT-X-KEY。如果播放列表仅有一个#EXT-X-KEY，则密钥文件的生命期从当前#EXT-X-KEY开始到播放列表结束；如果播放列表有两个或以上的#EXT-X-KEY，则密钥文件的生命期从当前#EXT-X-KEY开始到下一个#EXT-X-KEY结束。

METHOD属性为NONE时，表示媒体内容未加密，这种情况下不允许出现URI、IV、KEYFORMAT、KEYFORMATVERSIONS等属性；

METHOD属性为AES-128时，表示媒体内容采用AES-128方式对TS切片文件直接加密，这种情况下URI属性必须出现，IV属性可以出现也可以不出现；

IV为十六进制整数，代表加密初始向量。采用AES-128方式加密时，如果IV属性存在，则必须使用IV作为初始向量实现加密；如果IV属性不存在，使用媒体段的序列码作为初始向量实现加密。

如果属性METHOD的值为AES-128，并且播放列表中包含#EXT-X-I-FRAMES-ONLY标签（只包含I帧流），则整个媒体段使用AES-128 CBC加密。如果属性METHOD的值为AES-128，并且播放列表中不包含#EXT-X-I-FRAMES-ONLY标签，则CBC不能跨越媒体段，每个媒体段单独使用AES-128 CBC加密，从而实现整个媒体段全部内容的加密。

总的来说，流媒体客户端的解密模块需要先使用HTTP的GET请求下载#EXT-X-KEY标签的URI属性对应的密钥，然后根据METHOD属性指定的解密算法和IV属性的初始向量解密使用HTTP的GET请求下载的一个个TS片段。

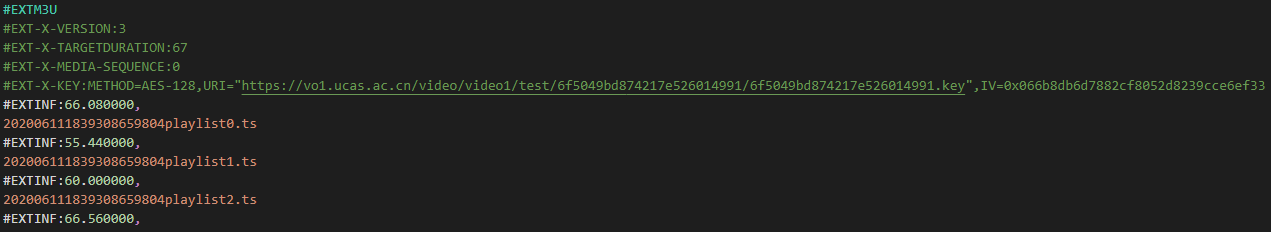
# 流媒体解密验证

用了HLS协议来验证流媒体解密。

## 4.1 自搭建环境验证流媒体解密

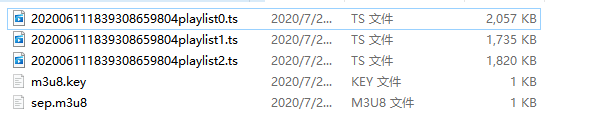
简单验证一下HLS流媒体协议的解密过程：

在一个视频网站下载原始的m3u8文件：



按照文件中指定的地址下载key文件和ts文件，初始的m3u8文件中指定了40个ts文件，为了方便只选择了3个文件（因为HLS协议对于所有的ts文件碎片的请求网络行为是相同的）。尝试用Windows上的视频播放器打开这三个ts文件，提示乱码无法打开。

对m3u8文件修改，把其中的ts文件数改为3个。最终的5个文件如下：



这5个文件：

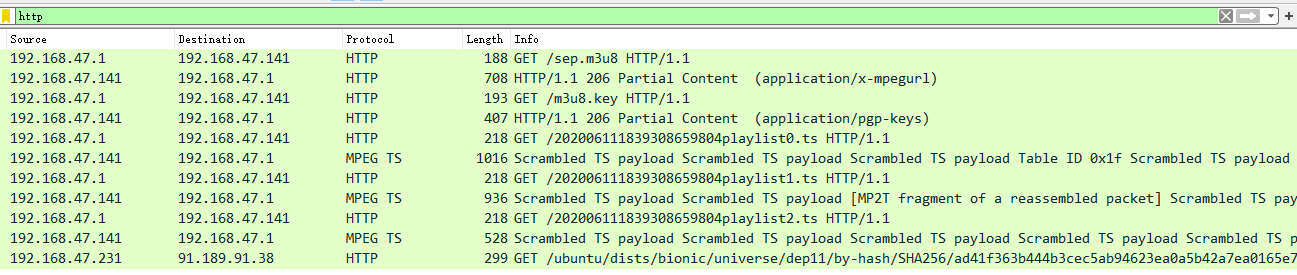
1. 环境搭建

开启的Linux上的apache2服务器（地址为192.168.47.141），把以上五个文件拷贝到/var/www/html文件夹下。对m3u8文件修改，把key文件的地址改为“http:/192.168.47.141/ m3u8.key”。

1. 开始抓包

在客户端（192.168.47.1）使用ffmpeg -i “http:/192.168.47.141/sep.m3u8” test.mp4命令，开始发起HTTP请求并开始下载，同时抓包。得到如下pcap包：

在wires hark中设置只显示http报文：



发现：ffmpeg依次请求了m3u8文件，key文件，ts文件。接着ffmpeg软件自动的解密了这三个ts文件，并把这三个ts文件合并成了test.mp4。使用Windows上的视频播放软件，发现这个视频可正常播放。

以上过程说明我们观察到了一次HLS流媒体协议的解密过程。

## 根据m3u8链接编写代码直接验证

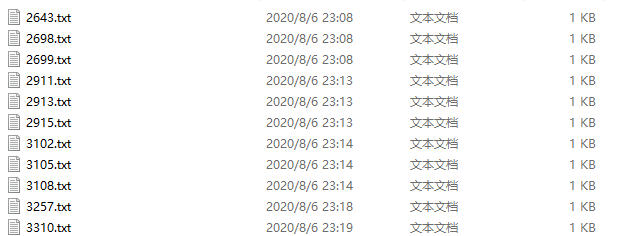
首先找到一批可用的m3u8链接，这些m3u8链接对应的m3u8文件都带有#EXT-X-KEY标签。其次下载这些m3u8链接对应的m3u8文件，得到一个可供验证的数据集：

这个数据集的文件夹结构为： ├── key

│   ├── m3u8

│   └── url

url文件夹下存储着469个txt文件，每个文件只有一行，是一个m3u8文件链接：



m3u8文件夹下也存储着469个txt文件，每个文件是一个m3u8文件。并且url文件夹下一个m3u8链接对应的m3u8文件就是m3u8文件夹下同名的txt文件。

这里使用了python语言构建验证程序，为了完整的模拟HLS会话过程，首先使用requests库下载m3u8库

|  |
| --- |
| try:      # 发起对m3u8文件的请求      key\_value = requests.get('http://' + urls[file\_count - BEGIN\_URL\_POS],                                  timeout=60)      if key\_value.status\_code != 200:          print("请求m3u8文件失败")          continue      print('此视频共有{}个ts文件片段'.format(ts\_count))  except BaseException:      # 网络连接有问题, 不处理这个      print("请求m3u8文件失败")      continue |

其次使用M3U8库提取m3u8文件的#EXT-X-KEY标签的各个属性：

|  |
| --- |
| import m3u8  m3u8\_obj = m3u8.load(os.path.join(KEY\_M3U8FILE\_DIR, file\_name))  # if 'huishenghuo888888' in urls[file\_count - BEGIN\_URL\_POS]:  #     continue  # 通常一个m3u8文件中只有一个key  key = m3u8\_obj.keys[0]  # 非加密的，或加密方法为None  if key is None:      continue  elif key.method is None or key.method.lower() == 'none':      continue  else:      # 若初始向量为空,则默认为全0      if key.iv is None:          key\_iv = b'0000000000000000'      else:          key\_iv = key.iv |

再其次使用使用requests库下载key文件：

|  |
| --- |
| try:      # 发起对key文件的请求      key\_value = requests.get(          'http://' + get\_real\_url(urls[file\_count - BEGIN\_URL\_POS].split('/'), key.uri),          timeout=60)      if key\_value.status\_code != 200:          print("请求KEY文件失败")          continue      key\_value = key\_value.content  except BaseException:      # 网络连接有问题, 不处理这个      print("请求KEY文件失败")      continue |

再其次使用asyncio库和aiohttp库并发的下载ts片段：

|  |
| --- |
| req\_info = namedtuple('Req\_info', 'req\_no req\_url')  req\_result = namedtuple('Req\_res', 'req\_no req\_status req\_status\_info req\_res')  HTTPStatus = Enum('Status', 'ok not\_found error')  # to\_be\_down\_urls是要下载的视频片段URL地址  to\_be\_down\_urls = []  # 要下载的视频片段数量  ts\_count = 0  for ts in m3u8\_obj.segments.by\_key(key):      ts\_count = ts\_count + 1      to\_be\_down\_urls.append(          get\_real\_url(urls[file\_count - BEGIN\_URL\_POS].split('/'), ts.uri))  # 构造下载请求列表  req\_info\_list = []  for i, line in enumerate(to\_be\_down\_urls[0:]):      req\_info\_list.append(          req\_info(str(i + 1),                      line[:-1] if line[-1] == '\n' else line))  # 开始下载  req\_result\_list = sorted(download\_many(req\_info\_list,                                          MAX\_REQ\_NUM),                              key=lambda x: int(x.req\_no))  def download\_many(req\_info\_list: list, max\_req\_num: list):      loop = asyncio.get\_event\_loop()      coro = downloader\_coro(req\_info\_list, max\_req\_num)      req\_result\_list = loop.run\_until\_complete(coro)      # loop.close()      return req\_result\_list |

最后使用Crypto库把下载到的一个个TS片段解密，拼接到一起，组成完整的文件：

|  |
| --- |
| # 依次解密TS片段，并在文件系统暂存解密后的TS片段  cipher = AES.new(key\_value, AES.MODE\_CBC, key\_iv)  plain\_seg = unpad(cipher.decrypt(r\_s.req\_res),                      AES.block\_size)  with open(DOWNLOAD\_PATH + str(file\_count) + '/' + r\_s.req\_no + '.ts', "wb") as seg\_ts\_file:      seg\_ts\_file.write(plain\_seg)  # 把在文件系统暂存的TS片段组合成一个完整的视频文件          plain = b''          ts\_file\_names = os.listdir(DOWNLOAD\_PATH + str(file\_count))          ts\_file\_count = []          for ts\_file\_name in ts\_file\_names:              ts\_file\_count.append(int(ts\_file\_name.split('.')[0]))          ts\_file\_count = sorted(ts\_file\_count)          for x in ts\_file\_count:              with open(DOWNLOAD\_PATH + str(file\_count) + '/' + str(x) + '.ts', 'rb') as seg\_ts\_file:                  plain += seg\_ts\_file.read()          with open(DOWNLOAD\_PATH + str(file\_count) + '.ts', 'wb') as ts\_file:              ts\_file.write(plain) |

最后，验证使用代码下载并解密的HLS流媒体文件可打开、可播放。验证了HLS协议定义的流媒体加密过程。

# 流媒体加密比例

## 北上广三地统计、三地的平均统计

|  |  |  |  |
| --- | --- | --- | --- |
|  | 总数 | 包含“EXT-X-KEY”的条数 | 比例 |
| 北京 | 37421 | 2419 | 6.5% |
| 上海 | 40889 | 3944 | 9.6% |
| 广州 | 50101 | 4280 | 8.5% |
| 总共 | 128411 | 10643 | 8.3% |

## https占比

从beijing\_media\_index\_data.txt和guangdong\_media\_index\_data.txt和shanghai\_media\_index\_data.txt三个文件中，共获取了2778871个URL。其中一共有64003条不同的URL，其中HTTP链接有48070条，HTTPS链接有15933条。HTTPS链接占比为：15933 / 64003 \* 100% = 24.894%。