Digital Transmission Licensing Administrator

DTCP Volume 1 Supplement K
DTCP-IP over HTTP Live Streaming
(Informational Version)

Hitachi Maxell, Ltd.
Intel Corporation
Panasonic Corporation
Sony Corporation
Toshiba Corporation

ARRIS Group, Inc., Contributor

Revision 1.0
June 9, 2017

Preface

Notice

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Hitachi Maxell, Ltd., Intel, Panasonic, Sony, and Toshiba (collectively, the "5C") disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in this Specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Some portions of this document, identified as "Draft" are in an intermediate draft form and are subject to change without notice. Adopters and other users of this Specification are cautioned these portions are preliminary, and that products based on it may not be interoperable with the final version or subsequent versions thereof.

Intellectual Property

Implementation of this Specification requires a license from the Digital Transmission Licensing Administrator.

Contact Information

Feedback on this Specification should be addressed to dtcp-services@dtcp.com.

Printing History:

TABLE OF CONTENTS

V1SK 1 Introduction

This supplement presents an adaptation of DTCP-IP protection to content streamed over HTTP Live Streaming (HLS). The scheme strives to employ as much of DTCP-IP's current definition as possible, recognizing that any adaptation must conform to the HLS specification or HLS-enabled device will not receive the flows. Therefore this supplement assumes that all aspects of DTCP-IP functionality such as the AKE process, content key creation, key lifetimes, and certificate processing are preserved as defined in the latest revision of DTCP Volume 1 Supplement E Mapping DTCP to IP. It makes use of HLS for encapsulation and the encryption algorithm details.

V1SK 1.1 Related Documents

This specification shall be used in conjunction with the following publications. When the publications are superseded by an approved revision, the revision shall apply.

- Digital Transmission Content Protection Specification volume 1 and volume 2 (latest revision)
- DTCP Volume 1 Supplement E Mapping DTCP to IP (latest revision)
- HTTP Live Streaming (HLS) draft-pantos-http-live-streaming (any revision)
- PKCS #7: Cryptographic Message Syntax Version 1.5

V1SK 1.2 Terms and Abbreviations

DTCP-IP	DTCP volume 1 Supplement E
HLS	HTTP Live Streaming
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IV	Initialization Vector
PCP	Protected Content Packet
PCKS7	Public Key Cryptography Standards #7
TS	MPEG-2 Transport Stream

V1SK 2 Modifications to Chapter 6 Content Channel Management Protection

This supplement provides a guideline for the source device to follow the DTCP-IP specification in performing AKE, creating PCPs and deriving an encryption key while constructing and transmitting encrypted HLS chunks using the same encryption key. Although HLS uses AES-128 CBC with IV for content encryption, similar to DTCP-IP, it employs a standard PCKS7 padding that differs from that of DTCP-IP. The sink device is assumed to be capable of decrypting HLS encrypted chunks using a key derived by the DTCP-IP sink device after the AKE process.

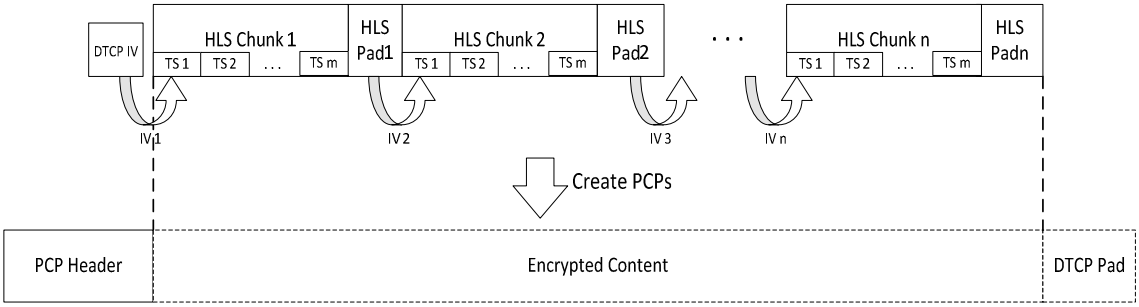
Typical HLS chunks are much shorter than typical PCPs. Even though it is possible to include one HLS chunk per each PCP, it is not recommended in this supplement. Thus it is more suitable to package multiple HLS chunks per PCP payload so that a single content key derived from the DTCP-IP key management algorithms could be assigned to a number of HLS chunks, as it would be so assigned to a single longer PCP in the current DTCP-IP definition. The equivalent PCP header fields as well as associated IV shall be supplied through the HLS KeyTag in the HLS playlist, in the form of a query field appended to the KeyTag URL as explained in section V1SK 2.2.

V1SK 2.1 Protected Content Packet (PCP)

The format of Protected Content Packets (PCP) carrying the PCP header and DTCP encrypted content (EC) remains the same as explained in section V1SE 4.26.1 of DTCP IP supplement – Ref #2, that starts with a PCP header followed by encrypted content and a padding. However the source device makes adjustments in the creation and packaging of encrypted content to produce encrypted HLS chunks arranged according to the HLS specification – Ref #3. Since encrypted HLS chunks are already padded using the PKCS7 standard, there is no need to add a DTCP pad at the end of the N HLS chunks that would roughly correspond to the encrypted content payload of a PCP. Standard HLS chunks are usually several seconds of MPEG-2 Transport Stream (TS), encrypted with AES-128 CBC. So the source device can view the HLS adaptation roughly as a repackaging of an encrypted PCP structure, protected under the identical key.

V1SK 2.1.1 Creating HLS chunks and PCPs

The source device shall create HLS chunks from clear MPEG-2 TS data blocks with desired chunk duration. Then each chunk is encrypted using AES CBC with IV with added PCKS7 pad.



Above picture shows the process of creating PCPs with HLS chunks:

- The source device shall create HLS chunks by concatenating TS packets to predefined chunk duration ensuring each chunk ends with complete TS packet. Therefore chunks are always 188x bytes aligned.
- The source device shall use the initial DTCP IV as the IV to encrypt the first chunk. The IV for all subsequent chunks shall be determined by good cryptographic practices as long as the IV is signaled in the HLS playlist accordingly. One can chose to take any (or the last) 16 bytes of the cipher text from the previous chunk or create the IV of the next chunk completely randomly.
- The source device shall add padding to the end of HLS chunk as needed per PCKS7 standard.
- The source device shall derive DTCP-IP key to encrypt each chunk using the IV stated above.
- The source device keeps track of the IV that is used for each HLS chunk for later inclusion in the HLS playlist file.

Note: A source device may avoid extra DTCP zero padding at the end of the last HLS chunk of the "PCP" by aligning HLS chunks with 752-byte blocks of TS data. Since the least common multiple of 16 and 188 is 752, this makes each HLS chunk a multiple of 188 bytes (to satisfy TS packet boundary size) and 16 bytes (to satisfy AES block cipher requirement size) causing the DTCP "PCP" to have no DTCP pad. Therefore there will only be 16 bytes of HLS PKCS7 pad at the end of each chunk. There is actually no need for the source device to encapsulate HLS chunks into actual PCP payloads with DTCP pads as described in V1SE 4.26.1 of DTCP IP supplement – Ref #2, since the sink device receives HLS chunks directly as described in the HLS playlist. However the PCP header is needed to derive a proper DTCP decryption key.

V1SK 2.1.2 Signaling PCP header in HLS manifest

The source device appends the original PCP header information to the HLS playlist KeyTag URL, so that it can be provided to the sink device for key derivation. Of the 14 bytes of PCP header, the last 4 length bytes are not needed since the total size of all chunks included in the PCP is not determined in cases such as Live Streaming. Therefore the **?pcph** field shall be only the first 10 bytes of the DTCP PCP header, suitably base-64 encoded. The source device also includes IV per each HLS chunk at the end of key tag URL as **,IV=**. Here is one example:

> #EXT-X-KEY:METHOD=AES-128,URI="dtcphls://<MediaServer>/hlskey**?pcph**=AkRCAAKVyRMZZg==",**IV=**0xd7dd62914a57a1a0d3f12a05a6885b1a

Where <MediaServer> is the domain name of the source device streaming HLS content.

The typical HLS playlist would be the real-time rolling style playlist, with 3 HLS chunks defined. Thus real-time streaming would be supported. Each content chunk in the playlist would have a corresponding KeyTag, as the IV changes for each chunk as described above. While rolling through a many minutes long PCP, only one actual **?pcph** query field would be in use, as the chunks all come from the same notional PCP and are encrypted with the same key. Only when the end of one rolling style playlist was reached, and a second one started, would there be two different **?pcph** fields, and two different keys, present in the playlist and in use for a brief time.

V1SK 2.2 Protected Content Packet 2 Format

This supplement does not expose any changes to PCP2 format as it is a slightly different packet format meant to allow the use of CMI descriptors in key derivation and rules processing. The ?pcph field added to HLS KeyTag URL as described above needs no changes, and it can carry either PCP or PCP2 as a packet type field within the PCP header to indicate whether the header format is PCP or PCP2.

V1SK 2.2.1 Signaling PCP2 and CMI in HLS manifest

This supplement adds another query string field, called "cmi", the equivalent of the "CMI field" of DTCP, which is a concatenation of all the CMI descriptors to be sent by the source device. Since this data is going to be carried over KeyTag URL, this cmi field must be base 64 encoded.

For example, with CMI descriptor 0, 1 and 2 defined and carried, the DTCP CMI field would have 17 bytes, and a base 64 coded version would have 23 characters:

> #EXT-X-KEY:METHOD=AES-128,URI="dtcphls://<MediaServer>/hlskey?pcph=AkRCAAKVyRMZZg&cmi=AAAAABAABAADKVyRMZZghFm=",IV=0xd7dd62914a57a1a0d3f12a05a6885b1a
>
> Note: The cmi field above does not correspond to any specific cmi data, but is present to show format and length.

V1SK 3 HLS Playlist GET and Playlist Example

The HTTP GET of the HLS playlist issued by the client has the following example HTTP header structure:

GET /directoryname/Movie.m3u8

Host: arrismediaserver:7878

Accept: */*

User-Agent: ABCProxy/0.1 (iPad; en_us)

An example HLS playlist, Movie.m3u8, has the following structure, assuming approximately 2 second duration chunks, and shows the first 3 chunks of the content:

```
#EXTM3U
#EXT-X-TARGETDURATION: 2
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-KEY:METHOD=AES-
128,URI="dtcphls://arrismediaserver/hlskey?pcph=AkRCAAKVyRMZZg==",IV=0xd7dd62914a57a1a
0d3f12a05a6885b1a
#EXTINF:2,
Movie_0.ts
#EXT-X-KEY:METHOD=AES-128,URI="
dtcphls://arrismediaserver/hlskey?pcph=AkRCAAKVyRMZZg==",IV=0x4f755cc6fe00c7f69df46a96
03f60aab
#EXTINF:2,
Movie_1.ts
#EXT-X-KEY:METHOD=AES-128,URI="
dtcphls://arrismediaserver/hlskey?pcph=AkRCAAKVyRMZZg==",IV=0x3c1f507b6806a33c1e1a544e
1eff1e86
#EXTINF:2,
Movie_2.ts
```

For the above example, the following attributes were used:

The 10 byte PCP header is base 64 encoded in the HLS KeyTag "?pcph" query string. Thus 10 bytes become 16 as shown. The PCP header fields in this example [bytes 0 and 1] are for copy free content with redistribution control, and exchange key label 0x44, and baseline AES cipher. As required by DTCP-IP, the first nonce field [header bytes 2-9] of any connection starts with the "PCP-UR" field, header bytes 2 and 3, and ends with a 48 bit field SNC whose MS-bit is a zero. The remaining 47 bits are random. Subsequent SNC fields (from subsequent PCP packets) within the same content flow can increment as required. The 3 chunks above are the first 3 2 second portions of what would have been the first PCP packet of a connection flow. Thus the 3 KeyTags show 3 different IVs but the same PCP header, and thus indicate the same content key.

Once the content flow moved on to what would have been the second PCP packet (and second PCP header), a different ?pcph field would appear, indicating a second key was in use for the matching chunks.

Note that this example used an https proxy to deliver key.

The advantage of these techniques are that they still draw on known standards, DLNA, DTCP, and HLS, although in a unique manner, and they meet all the key lifetime obligations of DTCP.