

# **High-bandwidth Digital Content Protection System**

## **Mapping HDCP to WHDI**

Revision 2.3

24 July, 2018

## 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. Intel Corporation disclaims 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.

The cryptographic functions described in this specification may be subject to export control by the United States, Japanese, and/or other governments.

Copyright © 1999-2008 by Intel Corporation. Third-party brands and names are the property of their respective owners.

## Acknowledgement

Amimon Inc. has contributed to the development of this specification.

## Intellectual Property

Implementation of this specification requires a license from the Digital Content Protection LLC.

### Contact Information

Digital Content Protection LLC  
C/O Vital Technical Marketing, Inc.  
3855 SW 153rd Drive  
Beaverton, OR 97006

Email: [info@digital-cp.com](mailto:info@digital-cp.com)

Web: [www.digital-cp.com](http://www.digital-cp.com)

## Revision History

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Scope	5
1.2	Definitions	5
1.3	Overview	8
1.4	Terminology	9
1.5	Changes from previous specification	10
1.6	References	10
<b>2</b>	<b>Authentication Protocol</b>	<b>11</b>
2.1	Overview	11
2.2	Authentication and Key Exchange	12
2.2.1	Pairing	17
2.3	Locality Check	18
2.4	Session Key Exchange	21
2.5	Authentication with Repeaters	21
2.5.1	Upstream Propagation of Topology Information	22
2.5.2	Downstream Propagation of Content Stream Management Information	28
2.6	Link Synchronization	29
2.7	Key Derivation	30
2.8	HDCP Transmitter State Diagram	30
2.9	HDCP Receiver State Diagram	36
2.10	HDCP Repeater State Diagrams	37
2.10.1	Propagation of Topology Errors and AVReady / AVUnready Notification	38
2.10.2	HDCP Repeater Downstream State Diagram	39
2.10.3	HDCP Repeater Upstream State Diagram	44
2.11	Converters	49
2.11.1	HDCP 2 – HDCP 1.x Converters	49
2.11.2	HDCP 1.x – HDCP 2 Converters	51
2.12	Session Key Validity	52
2.13	Random Number Generation	53
<b>3</b>	<b>HDCP Encryption</b>	<b>54</b>
3.1	Description	54
3.2	AV Stream	54
3.3	HDCP Cipher	54
3.3.1	HDCP Cipher for Coarse Stream	54
3.4	HDCP Encryption Indication	56
3.5	Uniqueness of $k_s$ and $r_{iv}$	58
<b>4</b>	<b>Authentication Protocol Messages</b>	<b>60</b>
4.1	HDCP Messages in WHDI	60
4.2	Message Format	61
4.2.1	AKE_Init (Transmitter to Receiver)	61
4.2.2	AKE_Send_Cert (Receiver to Transmitter)	61
4.2.3	AKE_No_Stored_km (Transmitter to Receiver)	61
4.2.4	AKE_Stored_km (Transmitter to Receiver)	61
4.2.5	AKE_Send_rrx (Receiver to Transmitter)	61
4.2.6	AKE_Send_H_prime (Receiver to Transmitter)	62
4.2.7	AKE_Send_Pairing_Info (Receiver to Transmitter)	62
4.2.8	LC_Init (Transmitter to Receiver)	62
4.2.9	LC_Send_L_prime (Receiver to Transmitter)	62
4.2.10	SKE_Send_Eks (Transmitter to Receiver)	62
4.2.11	RepeaterAuth_Send_ReceiverID_List (Receiver to Transmitter)	63
4.2.12	RTT_Ready (Receiver to Transmitter)	64
4.2.13	RTT_Challenge (Transmitter to Receiver)	64

4.2.14	Receiver_AuthStatus (Receiver to Transmitter).....	64
4.2.15	AKE_Transmitter_Info (Transmitter to Receiver).....	65
4.2.16	AKE_Receiver_Info (Receiver to Transmitter).....	65
4.2.17	RepeaterAuth_Send_Ack (Transmitter to Receiver) .....	66
4.2.18	RepeaterAuth_Stream_Manage (Transmitter to Receiver) .....	66
4.2.19	RepeaterAuth_Stream_Ready (Transmitter to Receiver) .....	67
<b>5</b>	<b>Renewability.....</b>	<b>68</b>
5.1	SRM Size and Scalability.....	69
5.2	Updating SRMs .....	70
<b>Appendix A.</b>	<b>Confidentiality and Integrity of Values.....</b>	<b>72</b>
<b>Appendix B.</b>	<b>DCP LLC Public Key.....</b>	<b>75</b>

## 1 Introduction

### 1.1 Scope

This specification describes the mapping of High-bandwidth Digital Content Protection (HDCP) system to Wireless Home Digital Interface (WHDI), Revision 2.2.

For the purpose of this specification, it is assumed that the Audiovisual content is transmitted over a WHDI based wireless display link. In an HDCP System, two or more HDCP Devices are interconnected through an HDCP-protected Interface. The Audiovisual Content flows from the Upstream Content Control Function into the HDCP System at the most upstream HDCP Transmitter. From there the Audiovisual Content encrypted by the HDCP System, referred to as HDCP Content, flows through a tree-shaped topology of HDCP Receivers over HDCP-protected Interfaces. This specification describes a content protection mechanism for: (1) authentication of HDCP Receivers to their immediate upstream connection (i.e., an HDCP Transmitter), (2) revocation of HDCP Receivers that are determined by the Digital Content Protection, LLC, to be invalid, and (3) HDCP Encryption of Audiovisual Content over the HDCP-protected Interfaces between HDCP Transmitters and their downstream HDCP Receivers. HDCP Receivers may render the HDCP Content in audio and visual form for human consumption. HDCP Receivers may be HDCP Repeaters that serve as downstream HDCP Transmitters emitting the HDCP Content further downstream to one or more additional HDCP Receivers.

Unless otherwise specified, the term “HDCP Receiver” is also used to refer to the upstream HDCP-protected interface port of an HDCP Repeater. Similarly, the term “HDCP Transmitter” is also used to refer to the downstream HDCP-protected interface port of an HDCP Repeater

Except when specified otherwise, HDCP 2.3-compliant Devices must interoperate with other device compliant with HDCP2.3 and lower that are connected to their HDCP-protected Interface Ports using the same protocol. HDCP Transmitters must support HDCP Repeaters.

The state machines in this specification define the required behavior of HDCP Devices. The link-visible behavior of HDCP Devices implementing the specified state machines must be identical, even if implementations differ from the descriptions. The behavior of HDCP Devices implementing the specified state machines must also be identical from the perspective of an entity outside of the HDCP System.

Implementations must include all elements of the content protection system described herein, unless the element is specifically identified as informative or optional. Adopters must also ensure that implementations satisfy the robustness and compliance rules described in the HDCP license agreement.

Device discovery and association, and link setup and teardown, is outside the scope of this specification.

### 1.2 Definitions

The following terminology, as used throughout this specification, is defined as herein:

**Audiovisual Content.** Audiovisual works (as defined in the United States Copyright Act as in effect on January 1, 1978), text and graphic images, are referred to as *AudioVisual Content*.

**Authorized Device.** An HDCP Device that is permitted access to HDCP Content is referred to as an *Authorized Device*. An HDCP Transmitter may test if a connected HDCP Receiver is an Authorized Device by successfully completing the following stages of the authentication protocol – Authentication and Key Exchange (AKE) and Locality check. If the authentication protocol

successfully results in establishing authentication, then the other device is considered by the HDCP Transmitter to be an Authorized Device.

**Content Stream** *Content Stream* consists of Audiovisual content received from an Upstream Content Control Function that is to be encrypted and Audiovisual content received from Upstream Content Control Function that is encrypted by the HDCP function.

**Device Key Set.** An HDCP Receiver has a Device Key Set, which consists of its corresponding Device Secret Keys along with the associated Public Key Certificate.

**Device Secret Keys.** For an HDCP Transmitter, Device Secret Key consists of the secret global constant. For an HDCP Receiver, Device Secret Keys consists of the secret global constant and the RSA private key. The Device Secret Keys are to be protected from exposure outside of the HDCP Device.

**downstream.** The term, *downstream*, is used as an adjective to refer to being towards the sink of the HDCP Content stream. For example, when an HDCP Transmitter and an HDCP Receiver are connected over an HDCP-protected Interface, the HDCP Receiver can be referred to as the *downstream* HDCP Device in this connection. For another example, on an HDCP Repeater, the HDCP-protected Interface Port(s) which can emit HDCP Content can be referred to as its *downstream* HDCP-protected Interface Port(s). See also, *upstream*.

**frame.** For purposes of the application of HDCP to WHDI, a frame consists of one video frame unit which also corresponds to one WHDI frame.

**Global Constant.** A 128-bit random, secret constant provided only to HDCP Adopters and used during HDCP Content encryption or decryption

**HDCP 1.x.** *HDCP 1.x* refers to, specifically, the variant of HDCP described by Revision 1.00 (referred to as HDCP 1.0), Revision 1.10 (referred to as HDCP 1.1), Revision 1.20 (referred to as HDCP 1.2) and Revision 1.30 (referred to as HDCP 1.3) along with their associated errata, if applicable.

**HDCP 1.x-compliant Device.** An HDCP Device that is designed in adherence to HDCP 1.x, defined above, is referred to as an *HDCP 1.x-compliant Device*.

**HDCP 2.** *HDCP 2* refers to, specifically, the variant of HDCP mapping for all HDCP protected interfaces (including WHDI) described by Revision 2.00 and higher versions along with their associated errata, if applicable.

**HDCP 2.0.** *HDCP 2.0* refers to, specifically, the variant of HDCP mapping described by Revision 2.00 of this specification along with its associated errata, if applicable.

**HDCP 2.0-compliant Device.** An HDCP Device that is designed in adherence to HDCP 2.0 is referred to as an *HDCP 2.0-compliant Device*.

**HDCP 2.3** *HDCP 2.3* refers to, specifically, the variant of HDCP mapping described by Revision 2.30 of this specification along with its associated errata, if applicable.

**HDCP 2.3-compliant Device.** An HDCP Device that is designed in adherence to HDCP 2.3 is referred to as an *HDCP 2.3-compliant Device*.

**HDCP Content.** *HDCP Content* consists of Audiovisual Content that is protected by the HDCP System. *HDCP Content* includes the Audiovisual Content in encrypted form as it is transferred from

an HDCP Transmitter to an HDCP Receiver over an HDCP-protected Interface, as well as any translations of the same content, or portions thereof. For avoidance of doubt, Audiovisual Content that is never encrypted by the HDCP System is not *HDCP Content*.

**HDCP Device.** Any device that contains one or more HDCP-protected Interface Port and is designed in adherence to HDCP is referred to as an *HDCP Device*.

**HDCP Encryption.** *HDCP Encryption* is the encryption technology of HDCP when applied to the protection of HDCP Content in an HDCP System.

**HDCP Receiver.** An HDCP Device that can receive and decrypt HDCP Content through one or more of its HDCP-protected Interface Ports is referred to as an *HDCP Receiver*.

**HDCP Repeater.** An HDCP Device that can receive and decrypt HDCP Content through one or more of its HDCP-protected Interface Ports, and can also re-encrypt and emit said HDCP Content through one or more of its HDCP-protected Interface Ports, is referred to as an *HDCP Repeater*. An *HDCP Repeater* may also be referred to as either an HDCP Receiver or an HDCP Transmitter when referring to either the upstream side or the downstream side, respectively.

**HDCP Session.** An *HDCP Session* is established between an HDCP Transmitter and HDCP Receiver with the transmission or reception of  $r_{tx}$  as part of the authentication initiation message, AKE\_Init. The established HDCP Session remains valid until it is aborted by the HDCP Transmitter or a new HDCP Session is established, which invalidates the HDCP Session that was previously established, by the transmission or reception of a new  $r_{tx}$  as part of the AKE\_Init message.

**HDCP System.** An *HDCP System* consists of an HDCP Transmitter, zero or more HDCP Repeaters and one or more HDCP Receivers connected through their HDCP-protected interfaces in a tree topology; whereas the said HDCP Transmitter is the HDCP Device most upstream, and receives the Audiovisual Content from an Upstream Content Control Function. All HDCP Devices connected to other HDCP Devices in an *HDCP System* over HDCP-protected Interfaces are part of the *HDCP System*.

**HDCP Transmitter.** An HDCP Device that can encrypt and emit HDCP Content through one or more of its HDCP-protected Interface Ports is referred to as an *HDCP Transmitter*.

**HDCP.** *HDCP* is an acronym for High-bandwidth Digital Content Protection. This term refers to this content protection system as described by any revision of this specification and its errata.

**HDCP-protected Interface Port.** A logical connection point on an HDCP Device that supports an HDCP-protected Interface is referred to as an *HDCP-protected Interface Port*. A single wireless connection can be made over an HDCP-protected interface port.

**HDCP-protected Interface.** An interface for which HDCP applies is described as an *HDCP-protected Interface*.

**Master Key.** A 128-bit random, secret cryptographic key negotiated between the HDCP Transmitter and the HDCP Receiver during Authentication and Key Exchange and used to pair the HDCP Transmitter with the HDCP Receiver.

**Public Key Certificate.** Each HDCP Receiver is issued a Public Key Certificate signed by DCP LLC, and contains the Receiver ID and RSA public key corresponding to the HDCP Receiver.

**Receiver ID.** A 40-bit value that uniquely identifies the HDCP Receiver. It has the same format as an HDCP 1.x KSV i.e. it contains 20 ones and 20 zeroes.

**Session Key.** A 128-bit random, secret cryptographic key negotiated between the HDCP Transmitter and the HDCP Receiver during Session Key Exchange and used during HDCP Content encryption and decryption.

**Upstream Content Control Function.** The HDCP Transmitter most upstream in the HDCP System receives Audiovisual Content to be protected from the *Upstream Content Control Function*. An instance of the *Upstream Content Control Function* transmits a content stream to the HDCP Transmitter. The *Upstream Content Control Function* is not part of the HDCP System, and the methods used, if any, by the *Upstream Content Control Function* to determine for itself the HDCP System is correctly authenticated or permitted to receive the Audiovisual Content, or to transfer the Audiovisual Content to the HDCP System, are beyond the scope of this specification. On a personal computer platform, an example of an *Upstream Content Control Function* may be software designed to emit Audiovisual Content to a display or other presentation device that requires HDCP.

**upstream.** The term, *upstream*, is used as an adjective to refer to being towards the source of the HDCP Content stream. For example, when an HDCP Transmitter and an HDCP Receiver are connected over an HDCP-protected Interface, the HDCP Transmitter can be referred to as the *upstream* HDCP Device in this connection. For another example, on an HDCP Repeater, the HDCP-protected Interface Port(s) which can receive HDCP Content can be referred to as its *upstream* HDCP-protected Interface Port(s). See also, *downstream*.

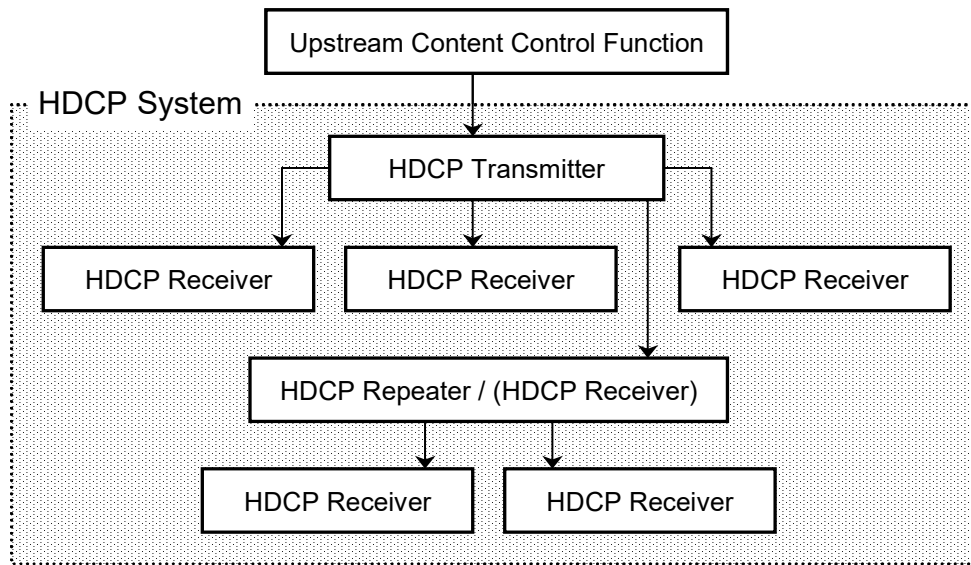
### 1.3 Overview

HDCP is designed to protect the transmission of Audiovisual Content between an HDCP Transmitter and an HDCP Receiver. The HDCP Transmitter may support simultaneous connections to HDCP Receivers through one or more of its HDCP-protected interface ports. The system also allows for HDCP Repeaters that support downstream HDCP-protected Interface Ports. The HDCP System places the following constraints on the number of HDCP Devices and levels of HDCP Repeaters in the topology.

1. Up to four levels of HDCP Repeaters and as many as 32 total HDCP Devices, including HDCP Repeaters, are allowed to be connected to an HDCP-protected Interface port; and
2. An instance of an Upstream Content Control Function transmits a content stream to the HDCP Transmitter. For every such content stream received and encrypted by the HDCP System, the HDCP Transmitter is allowed to transmit the generated HDCP Content stream to up to four levels of HDCP Repeaters and as many as 32 total HDCP Devices, including HDCP Repeaters.

Figure 1.1 illustrates an example connection topology for HDCP Devices.





**Figure 1.1. Sample Connection Topology of an HDCP System**

There are three elements of the content protection system. Each element plays a specific role in the system. First, there is the authentication protocol, through which the HDCP Transmitter verifies that a given HDCP Receiver is licensed to receive HDCP Content. The authentication protocol is implemented between the HDCP Transmitter and its corresponding downstream HDCP Receiver. With the legitimacy of the HDCP Receiver determined, encrypted HDCP Content is transmitted between the two devices based on shared secrets established during the authentication protocol. This prevents eavesdropping devices from utilizing the content. Finally, in the event that legitimate devices are compromised to permit unauthorized use of HDCP Content, renewability allows an HDCP Transmitter to identify such compromised devices and prevent the transmission of HDCP Content.

This document contains chapters describing in detail the requirements of each of these elements. In addition, a chapter is devoted to describing the cipher structure that is used in the encryption of HDCP Content.

## 1.4 Terminology

Throughout this specification, names that appear in *italic* refer to values that are exchanged during the HDCP cryptographic protocol. C-style notation is used throughout the state diagrams and protocol diagrams, although the logic functions AND, OR, and XOR are written out where a textual description would be more clear.

This specification uses the big-endian notation to represent bit strings so that the most significant bit in the representation is stored in the left-most bit position. The concatenation operator ‘||’ combines two values into one. For eight-bit values *a* and *b*, the result of (*a* || *b*) is a 16-bit value, with the value *a* in the most significant eight bits and *b* in the least significant eight bits.

## 1.5 Changes from previous specification

	Paragraph	Description
1	2.1	Protocol Descriptor Field in Public Key Certificate to differentiate between revision versions.
2	2.2	Modified H and H' calculation during AKE process without stored km
3	2.2.1	Ekh(km) Computation
4	2.3	<b>Locality check</b>
5	2.4	<b>Session Key Exchange</b>
6	2.7	<b>Key derivation</b>
7	2.8	HDCP Transmitter state diagram
8	2.10.2	HDCP Repeater State Diagram
9	4	Authentication Protocol Messages

## 1.6 References

- [1]. Digital Content Protection (DCP) LLC, High-bandwidth Digital Content Protection System, Revision 1.3, December 21, 2006.
- [2]. Digital Content Protection (DCP) LLC, HDCP Specification 1.3 – Amendment for DisplayPort, Revision 1.0, December 19, 2006.
- [3]. National Institute of Standards and Technology (NIST), *Advanced Encryption Standard (AES)*, FIPS Publication 197, November 26, 2001.
- [4]. RSA Laboratories, *RSA Cryptography Standard*, PKCS #1 v2.1, June 14, 2002.
- [5]. National Institute of Standards and Technology (NIST), *Secure Hash Standard (SHS)*, FIPS Publication 180-2, August 1, 2002.
- [6]. Internet Engineering Task Force (IETF), *HMAC: Keyed-Hashing for Message Authentication*, Request for Comments (RFC) 2104, February 1997.
- [7]. Wireless Home Digital Interface (WHDI) Specification, Revision 1.0.
- [8]. National Institute of Standards and Technology (NIST), Recommendation for Random Number Generation Using Deterministic Random Bit Generators, Special Publication 800-90, March 2007

## 2 Authentication Protocol

### 2.1 Overview

The HDCP Authentication protocol is an exchange between an HDCP Transmitter and an HDCP Receiver that affirms to the HDCP Transmitter that the HDCP Receiver is authorized to receive HDCP Content. It is comprised of the following stages

- Authentication and Key Exchange (AKE) – The HDCP Receiver’s public key certificate is verified by the HDCP Transmitter. A Master Key  $k_m$  is exchanged.
- Locality Check – The HDCP Transmitter enforces locality on the content by requiring that the Round Trip Time (RTT) between a pair of messages is not more than 7 ms.
- Session Key Exchange (SKE) – The HDCP Transmitter exchanges Session Key  $k_s$  with the HDCP Receiver.
- Authentication with Repeaters – The step is performed by the HDCP Transmitter only with HDCP Repeaters. In this step, the repeater assembles downstream topology information and forwards it to the upstream HDCP Transmitter.

Successful completion of AKE and locality check stages affirms to the HDCP Transmitter that the HDCP Receiver is authorized to receive HDCP Content. At the end of the authentication protocol, a communication path is established between the HDCP Transmitter and HDCP Receiver that only Authorized Devices can access.

All HDCP Devices contain a 128-bit secret global constant denoted by  $lc_{128}$ . All HDCP Devices share the same global constant.  $lc_{128}$  is provided only to HDCP Adopters.

The HDCP Transmitter contains the 3072-bit RSA public key of DCP LLC denoted by  $kpub_{dep}$ .

The HDCP Receiver is issued 1024-bit RSA public and private keys. The public key is stored in a Public Key Certificate issued by DCP LLC, denoted by  $cert_{rx}$ . Table 2.1 gives the fields contained in the certificate. All values are stored in big-endian format.

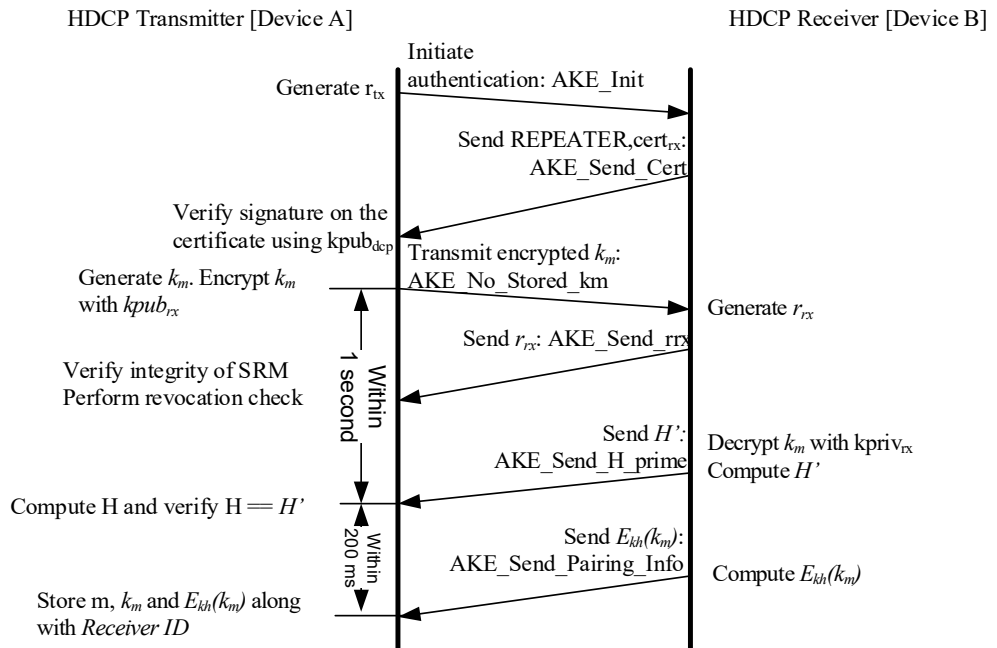
Name	Size (bits)	Bit position	Function
Receiver ID	40	4175:4136	Unique receiver identifier. It has the same format as an HDCP 1.x KSV i.e. it contains 20 ones and 20 zeroes
Receiver Public Key	1048	4135:3088	Unique RSA public key of HDCP Receiver denoted by $kpub_{rx}$ . The first 1024 bits is the big-endian representation of the modulus $n$ and the trailing 24 bits is the big-endian representation of the public exponent $e$
Protocol Descriptor	4	3087:3084	Protocol descriptor field. Possible values are 0x0 and 0x1. 0x2-0xF – Reserved for future use.
Reserved	12	3083:3072	Reserved for future definition. Must be 0x0000
DCP LLC Signature	3072	3071:0	A cryptographic signature calculated over all preceding fields of the certificate. RSASSA-PKCS1-v1_5 is the signature scheme used as defined by PKCS #1 V2.1: RSA Cryptography Standard. SHA-256 is the underlying hash function

**Table 2.1. Public Key Certificate of HDCP Receiver**

The secret RSA private key is denoted by  $k_{priv_{rx}}$ . The computation time of RSA private key operation can be reduced by using the Chinese Remainder Theorem (CRT) technique. Therefore, it is recommended that HDCP Receivers use the CRT technique for private key computations.

## 2.2 Authentication and Key Exchange

Authentication and Key Exchange (AKE) is the first step in the authentication protocol. Figure 2.1 and Figure 2.2 illustrates the AKE. The HDCP Transmitter (*Device A*) can initiate authentication at any time, even before a previous authentication exchange has completed. The HDCP Transmitter initiates a new HDCP Session by sending a new  $r_{tx}$  as part of the authentication initiation message, AKE\_Init. Message formats are defined in Section 4.2. The HDCP Transmitter must ignore out of sequence authentication protocol messages received from the HDCP Receiver.



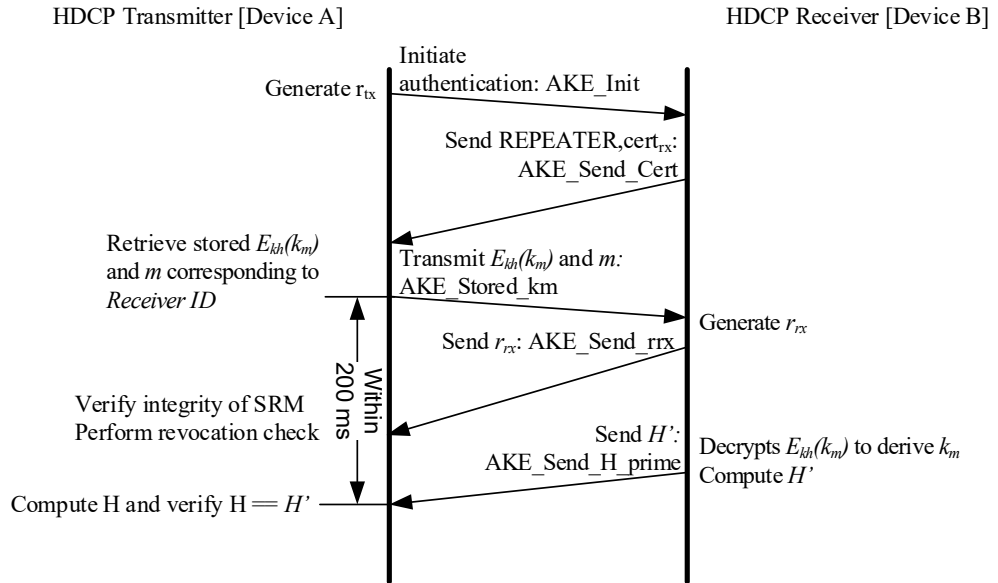
**Figure 2.1. Authentication and Key Exchange (Without Stored  $k_m$ )**

```

If (Protocol Descriptor = 0x1)
    H = HMAC-SHA256( $r_{tx}$ , XOR REPEATER || AKE_Receiver_Info.VERSION ||
    AKE_Receiver_Info.RECEIVER_CAPABILITY_MASK ||
    AKE_Transmitter_Info.VERSION || AKE_Transmitter_Info.
    TRANSMITTER_CAPABILITY_MASK,  $k_d$ )
If (Protocol Descriptor = 0x0)
    H = HMAC-SHA256( $r_{tx}$ , XOR REPEATER,  $k_d$ )
    
```

```

If (Protocol Descriptor = 0x1) and (HDCP Transmitter not HDCP2.0-compliant and not HDCP2.1-
compliant)
    H' = HMAC-SHA256( $r_{tx}$ , XOR REPEATER || AKE_Receiver_Info.VERSION ||
    AKE_Receiver_Info.RECEIVER_CAPABILITY_MASK ||
    AKE_Transmitter_Info.VERSION || AKE_Transmitter_Info.
    TRANSMITTER_CAPABILITY_MASK,  $k_d$ )
If (Protocol Descriptor = 0x0) or (HDCP Transmitter is HDCP2.0-compliant or HDCP2.1-compliant)
    H' = HMAC-SHA256( $r_{tx}$ , XOR REPEATER,  $k_d$ )
    
```



**Figure 2.2. Authentication and Key Exchange (With Stored  $k_m$ )**

<p>If (Protocol Descriptor = 0x1)  <math>H = \text{HMAC-SHA256}(r_{tx}, \text{XOR REPEATER}    \text{AKE\_Receiver\_Info.VERSION}    \text{AKE\_Receiver\_Info.RECEIVER\_CAPABILITY\_MASK}    \text{AKE\_Transmitter\_Info.VERSION}    \text{AKE\_Transmitter\_Info.TRANSMITTER\_CAPABILITY\_MASK}, k_d)</math></p> <p>If (Protocol Descriptor = 0x0)  <math>H = \text{HMAC-SHA256}(r_{tx}, \text{XOR REPEATER}, k_d)</math></p>	<p>If (Protocol Descriptor = 0x1) and (HDCP Transmitter not HDCP2.0-compliant and not HDCP2.1-compliant)  <math>H' = \text{HMAC-SHA256}(r_{rx}, \text{XOR REPEATER}    \text{AKE\_Receiver\_Info.VERSION}    \text{AKE\_Receiver\_Info.RECEIVER\_CAPABILITY\_MASK}    \text{AKE\_Transmitter\_Info.VERSION}    \text{AKE\_Transmitter\_Info.TRANSMITTER\_CAPABILITY\_MASK}, k_d)</math></p> <p>If (Protocol Descriptor = 0x0) or (HDCP Transmitter is HDCP2.0-compliant or HDCP2.1-compliant)  <math>H' = \text{HMAC-SHA256}(r_{rx}, \text{XOR REPEATER}, k_d)</math></p>
--	---

The HDCP Transmitter

- Initiates authentication by sending the initiation message, AKE\_Init, containing a 64-bit pseudo-random value ( $r_{tx}$ ).
- Receives AKE\_Send\_Cert from the receiver containing REPEATER and  $cert_{rx}$  values. REPEATER indicates whether the connected receiver is an HDCP Repeater

Note: The HDCP transmitter may use mechanisms outside the scope of the HDCP Specification to determine whether the HDCP Receiver is an HDCP 2.0 compliant Device. If the HDCP Transmitter determines, using mechanisms outside the scope of the HDCP specification, that the HDCP Receiver is an HDCP 2.0-compliant Device, it need not send the AKE\_Transmitter\_Info message to the HDCP receiver.

- Extracts Receiver ID from  $cert_{rx}$ 
  - If the HDCP Transmitter does not have a 128-bit Master Key  $k_m$  stored corresponding to the Receiver ID (See Section 2.2.1)
    - Verifies the signature on the certificate using  $k_{pub\_dcp}$ . Failure of signature verification constitutes an authentication failure and the HDCP Transmitter aborts the authentication protocol (See Section **Error! Reference source not found.** on handling authentication failures).

- Generates a pseudo-random 128-bit Master Key  $k_m$ . Encrypts  $k_m$  with  $k_{pub_{rx}}$  ( $E_{k_{pub}}(k_m)$ ) and sends AKE\_No\_Stored\_km message to the receiver containing the 1024-bit  $E_{k_{pub}}(k_m)$ . RSAES-OAEP encryption scheme must be used as defined by PKCS #1 V2.1: RSA Cryptography Standard. SHA-256 is the underlying hash function. The mask generation function used is MGF1 which uses SHA-256 as its underlying hash function.
- Verifies integrity of the System Renewability Message (SRM). It does this by checking the signature of the SRM using  $k_{pub_{dep}}$ . Failure of this integrity check constitutes an authentication failure and causes the HDCP Transmitter to abort authentication protocol (See Section **Error! Reference source not found.** on handling authentication failures).

The top-level HDCP Transmitter checks to see if the *Receiver ID* of the connected device is found in the revocation list. If the *Receiver ID* of the connected HDCP Device is found in the revocation list, authentication fails and the authentication protocol is aborted (See Section **Error! Reference source not found.** on handling authentication failures). SRM integrity check and revocation check are performed only by the top-level HDCP Transmitter.

- Receives AKE\_Send\_rrx message from the receiver containing the 64-bit pseudo-random value ( $r_{rx}$ ).
- Performs key derivation as explained in Section 2.7 to generate 256-bit  $k_d$ .  $k_d = dkey_0 || dkey_1$ , where  $dkey_0$  and  $dkey_1$  are derived keys generated when  $ctr = 0$  and  $ctr = 1$  respectively.  $dkey_0$  and  $dkey_1$  are in big-endian order.
- Computes 256-bit  $H = \text{HMAC-SHA256}(r_{rx} \text{ XOR REPEATER} || \text{AKE\_Receiver\_Info.VERSION} || \text{AKE\_Receiver\_Info.RECEIVER\_CAPABILITY\_MASK} || \text{AKE\_Transmitter\_Info.VERSION} || \text{AKE\_Transmitter\_Info.Transmitter\_CAPABILITY\_MASK}, k_d)$  if the Protocol Descriptor field in  $cert_{rx}$  is equal to 0x1. HMAC-SHA256 is computed over  $r_{rx} \text{ XOR REPEATER}$  concatenated with AKE\_Receiver\_Info.VERSION, AKE\_Receiver\_Info.RECEIVER\_CAPABILITY\_MASK, AKE\_Transmitter\_Info.VERSION, AKE\_Transmitter\_Info.Transmitter\_CAPABILITY\_MASK and the key used for HMAC is  $k_d$ . REPEATER is XORed with the least significant byte of  $r_{rx}$ . All values are in big-endian order.

Computes 256-bit  $H = \text{HMAC-SHA256}(r_{rx} \text{ XOR REPEATER}, k_d)$  if the Protocol Descriptor field in  $cert_{rx}$  is equal to 0x0. HMAC-SHA256 is computed over  $r_{rx} \text{ XOR REPEATER}$  and the key used for HMAC is  $k_d$ . REPEATER is XORed with the least significant byte of  $r_{rx}$ .

- Receives AKE\_Send\_H\_prime message from the receiver containing the 256-bit  $H'$ . This message must be received within one second after sending  $E_{k_{pub}}(k_m)$  (AKE\_No\_Stored\_km) to the receiver. Authentication fails and the authentication protocol is aborted if the message is not received within one second or there is a mismatch

between H and  $H'$  (See Section **Error! Reference source not found.** on handling authentication failures).

- If the HDCP Transmitter has a 128-bit Master Key  $k_m$  stored corresponding to the *Receiver ID* (See Section 2.2.1)
  - Sends AKE\_Stored\_km message to the receiver with the 128-bit  $E_{kh}(k_m)$  and the 128-bit  $m$  corresponding to the *Receiver ID* of the HDCP Receiver
  - Verifies integrity of the System Renewability Message (SRM). It does this by checking the signature of the SRM using  $k_{pub_{dcp}}$ . Failure of this integrity check constitutes an authentication failure and causes the HDCP Transmitter to abort the authentication protocol.

The top-level HDCP Transmitter checks to see if the *Receiver ID* of the connected device is found in the revocation list. If the *Receiver ID* of the connected HDCP Device is found in the revocation list, authentication fails and the authentication protocol is aborted.

- Receives AKE\_Send\_rrx message from the receiver containing the 64-bit pseudo-random value ( $r_{rx}$ ) from the receiver.
- Performs key derivation as explained in Section 2.7 to generate 256-bit  $k_d$ .  $k_d = dkey_0 \parallel dkey_1$ , where  $dkey_0$  and  $dkey_1$  are derived keys generated when  $ctr = 0$  and  $ctr = 1$  respectively.  $dkey_0$  and  $dkey_1$  are in big-endian order.
- Computes 256-bit  $H = \text{HMAC-SHA256}(r_{rx} \text{ XOR REPEATER} \parallel \text{AKE\_Receiver\_Info.VERSION} \parallel \text{AKE\_Receiver\_Info.RECEIVER\_CAPABILITY\_MASK} \parallel \text{AKE\_Transmitter\_Info.VERSION} \parallel \text{AKE\_Transmitter\_Info.Transmitter\_CAPABILITY\_MASK}, k_d)$  if the Protocol Descriptor field in  $cert_{rx}$  is equal to 0x1. HMAC-SHA256 is computed over  $r_{rx} \text{ XOR REPEATER}$  concatenated with AKE\_Receiver\_Info.VERSION, AKE\_Receiver\_Info.RECEIVER\_CAPABILITY\_MASK, AKE\_Transmitter\_Info.VERSION, AKE\_Transmitter\_Info.Transmitter\_CAPABILITY\_MASK and the key used for HMAC is  $k_d$ . REPEATER is XORed with the least significant byte of  $r_{rx}$ . All values are in big-endian order.

Computes 256-bit  $H = \text{HMAC-SHA256}(r_{rx} \text{ XOR REPEATER}, k_d)$  if the Protocol Descriptor field in  $cert_{rx}$  is equal to 0x0. HMAC-SHA256 is computed over  $r_{rx} \text{ XOR REPEATER}$  and the key used for HMAC is  $k_d$ . REPEATER is XORed with the least significant byte of  $r_{rx}$ .

- Receives AKE\_Send\_H\_prime message from the receiver containing the 256-bit  $H'$ . This message must be received within 200 ms after sending the AKE\_Stored\_km message to the receiver. Authentication fails and the authentication protocol is aborted if the message is not received within 200 ms or there is a mismatch between H and  $H'$  (See Section **Error! Reference source not found.** on handling authentication failures).

The HDPC Receiver

- Sends AKE\_Send\_Cert message in response to AKE\_Init
- Generates and sends 64-bit  $r_{rx}$  as part of the AKE\_Send\_rrx message immediately after receiving either AKE\_No\_Stored\_km or AKE\_Stored\_km message from the transmitter.  $r_{rx}$  must be generated only after either AKE\_No\_Stored\_km or AKE\_Stored\_km message is received from the transmitter.

- If AKE\_No\_Stored\_km is received, the HDPC Receiver

- Decrypts  $k_m$  with  $k_{priv_{rx}}$  using RSAES-OAEP decryption scheme.
- Performs key derivation as explained in Section 2.7 to generate 256-bit  $k_d$ .  $k_d = dkey_0 || dkey_1$ , where  $dkey_0$  and  $dkey_1$  are derived keys generated when  $ctr = 0$  and  $ctr = 1$  respectively.  $dkey_0$  and  $dkey_1$  are in big-endian order.
- Computes 256-bit  $H' = \text{HMAC-SHA256}(r_{tx} \text{ XOR REPEATER} || \text{AKE\_Receiver\_Info.VERSION} || \text{AKE\_Receiver\_Info.RECEIVER\_CAPABILITY\_MASK} || \text{AKE\_Transmitter\_Info.VERSION} || \text{AKE\_Transmitter\_Info.Transmitter\_CAPABILITY\_MASK}, k_d)$  if the Protocol Descriptor field in  $cert_{rx}$  is equal to 0x1 and the HDPC Transmitter is not HDPC2.0-compliant and is not HDPC2.1-compliant (i.e. AKE\_Transmitter\_Info.VERSION is not equal to 0x01).

Computes 256-bit  $H' = \text{HMAC-SHA256}(r_{tx} \text{ XOR REPEATER}, k_d)$  if the Protocol Descriptor field in  $cert_{rx}$  is equal to 0x0 or the HDPC Transmitter is HDPC2.0-compliant or HDPC2.1-compliant. (i.e. AKE\_Transmitter\_Info.VERSION is equal to 0x01).

The HDPC receiver sends AKE\_Send\_H\_prime message immediately after computation of  $H'$  to ensure that the message is received by the transmitter within the specified one second timeout at the transmitter.

- If AKE\_Stored\_km is received, the HDPC Receiver

- Computes 128-bit  $k_h = \text{SHA-256}(k_{priv_{rx}})[127:0]$
- Decrypts  $E_{k_h}(k_m)$  using AES with the received  $m$  as input and  $k_h$  as key in to the AES module as illustrated in Figure 2.3 to derive  $k_m$ .
- Computes 256-bit  $H' = \text{HMAC-SHA256}(r_{tx} \text{ XOR REPEATER} || \text{AKE\_Receiver\_Info.VERSION} || \text{AKE\_Receiver\_Info.RECEIVER\_CAPABILITY\_MASK} || \text{AKE\_Transmitter\_Info.VERSION} || \text{AKE\_Transmitter\_Info.Transmitter\_CAPABILITY\_MASK}, k_d)$  if the Protocol Descriptor field in  $cert_{rx}$  is equal to 0x1 and the HDPC Transmitter is not HDPC2.0-compliant and is not HDPC2.1-compliant (i.e. AKE\_Transmitter\_Info.VERSION is not equal to 0x01).

Computes 256-bit  $H' = \text{HMAC-SHA256}(r_{tx} \text{ XOR REPEATER}, k_d)$  if the Protocol Descriptor field in  $cert_{rx}$  is equal to 0x0 or the HDPC



Transmitter is HDCP2.0-compliant or HDCP2.1-compliant. (i.e. AKE\_Transmitter\_Info.VERSION is equal to 0x01).

The HDCP receiver sends AKE\_Send\_H\_prime message immediately after computation of  $H'$  to ensure that the message is received by the transmitter within the specified 200 ms timeout at the transmitter.

On a decryption failure of  $k_m$  with  $k_{priv_{rx}}$ , the HDCP Receiver does not send  $H'$  and simply lets the timeout occur on the HDCP Transmitter.

## 2.2.1 Pairing

To speed up the AKE process, pairing must be implemented between the HDCP Transmitter and HDCP Receiver in parallel with AKE. When AKE\_No\_Stored\_km message is received from the transmitter, it is an indication to the receiver that the transmitter does not have  $k_m$  stored corresponding to the receiver. In this case, after computing  $H'$ , the HDCP Receiver

- Computes 128-bit  $k_h = \text{SHA-256}(k_{priv_{rx}}[127:0])$ .
- Generates 128-bit  $E_{kh}(k_m)$  by encrypting  $k_m$  with  $k_h$  using AES as illustrated in Figure 2.3.
- Sends AKE\_Send\_Pairing\_Info to the transmitter containing the 128-bit  $E_{kh}(k_m)$ .

On receiving AKE\_Send\_Pairing\_Info message, the HDCP Transmitter

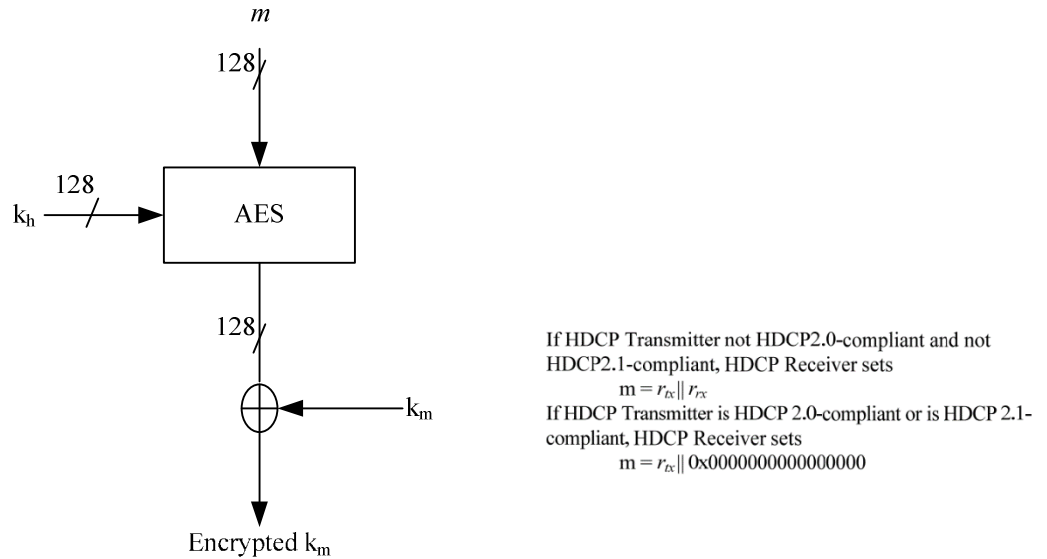
- Persistently stores  $m$  (which is  $r_{rx}$  appended with 64 0s),  $k_m$  and  $E_{kh}(k_m)$  along with Receiver ID.  $k_m$  and  $E_{kh}(k_m)$  must be stored securely.

If AKE\_Send\_Pairing\_Info is not received by the HDCP Transmitter within 200 ms of the reception of AKE\_Send\_H\_prime, authentication fails and the authentication protocol is aborted (See Section **Error! Reference source not found.** on handling authentication failures).

Note: The HDCP Transmitter must store in its non-volatile storage  $m$ ,  $k_m$  and  $E_{kh}(k_m)$  along with corresponding Receiver IDs of all HDCP Receivers with which pairing was implemented by the HDCP Transmitter.

If the HDCP receiver is HDCP 2.0 compliant or AKE\_Receiver\_Info.Version = 0x01, the HDCP Transmitter must not store pairing information ( $m$ ,  $k_m$ ,  $E_{kh}(k_m)$  and Receiver ID) corresponding to the receiver.

Figure 2.3 illustrates the encryption of  $k_m$  with  $k_h$ .



**Figure 2.3.  $E_{k_h}(k_m)$  Computation**

If the HDCP transmitter is not HDCP 2.0-compliant and is not HDCP 2.1 compliant (i.e. AKE\_Transmitter\_Info.VERSION is not equal to 0x01), the HDCP receiver constructs 128-bit  $m$  by concatenating  $r_{rx}$  with  $r_{rx}$  ( $r_{rx} || r_{rx}$ ). Both values in big-endian order.

If the HDCP transmitter is HDCP 2.0-compliant or HDCP 2.1 compliant (i.e. AKE\_Transmitter\_Info.VERSION is equal to 0x01), 128-bit  $m$  is constructed by the HDCP receiver by appending 64 0s to  $r_{rx}$ .  $r_{rx}$  is in big-endian order.

### 2.3 Locality Check

Locality check is performed after AKE and pairing. The HDCP Transmitter initiates locality check by sending a 64-bit pseudo-random nonce  $r_n$  to the downstream receiver.

If AKE\_Receiver\_Info.VERSION = 0x01 and the HDCP has set its TRANSMITTER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit to one in AKE\_Transmitter\_info message transmitted to the HDCP Receiver, the HDCP Transmitter must initiate re-authentication with the HDCP Receiver, the HDCP Transmitter must initiate re-authentication with the HDCP Receiver with the TRANSMITTER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit set to zero.

If the HDCP Receiver is HDCP 2.0 compliant or if the RECEIVER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit received as part of the AKE\_Receiver\_info message is set to zero or the transmitter has set the TRANSMITTER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit to zero in its AKE\_Transmitter\_info message,

The HDCP Transmitter

- Initiates locality check by sending LC\_Init message containing a 64-bit pseudo-random nonce  $r_n$  to the HDCP Receiver.

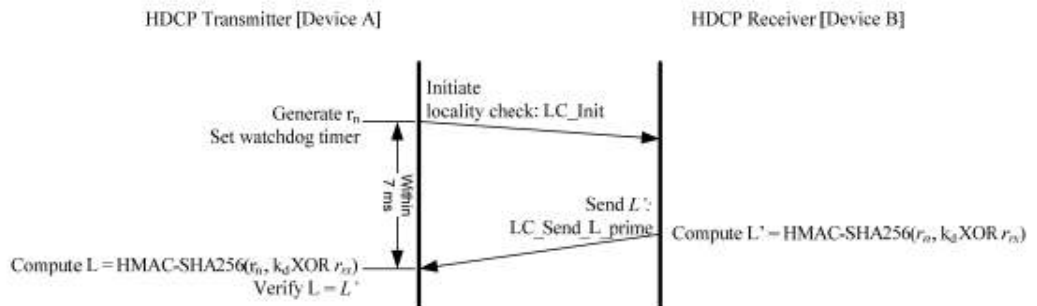
- Sets its watchdog timer to 7 ms. Locality check fails if the watchdog timer expires before LC\_Send\_L\_prime message is received.
- Computes  $L = \text{HMAC-SHA256}(r_n, k_d \text{ XOR } r_{rx})$  where HMAC-SHA256 is computed over  $r_n$  and the key used for HMAC is  $k_d \text{ XOR } r_{rx}$ , where  $r_{rx}$  is XORed with the least-significant 64-bits of  $k_d$ .
- On receiving LC\_Send\_L\_prime message, compares  $L$  and  $L'$ . Locality check fails if  $L$  is not equal to  $L'$ .

If the RECEIVER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit received as part of the AKE\_Receiver\_info message is set to one and the transmitter has set the TRANSMITTER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit to one in its AKE\_Transmitter\_info message, the HDCP transmitter:

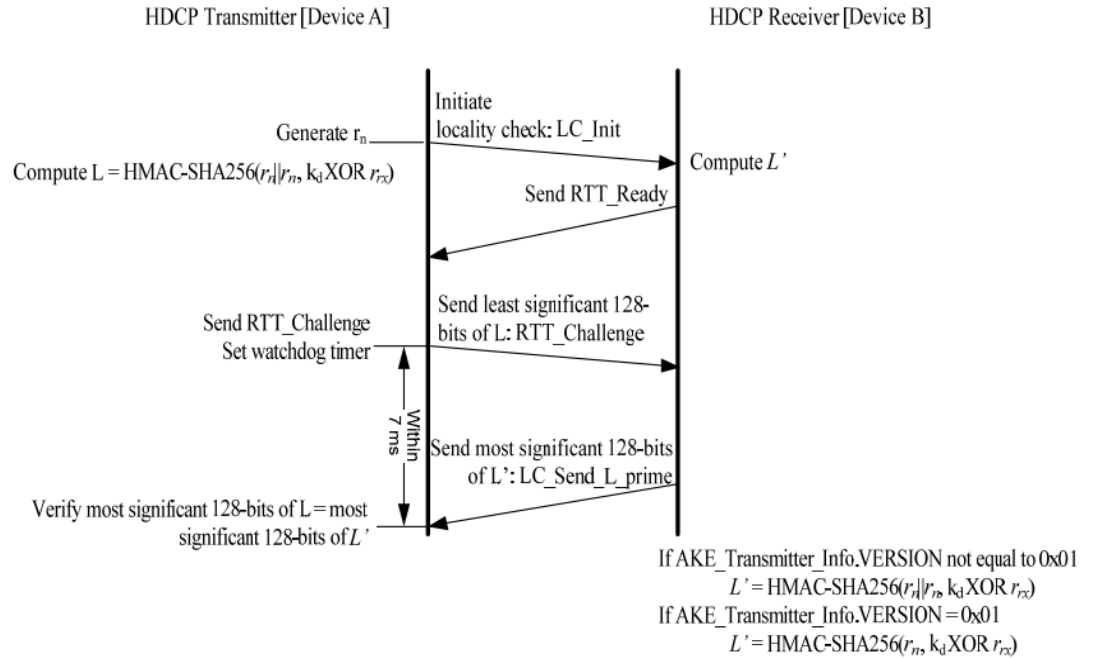
- Initiates locality check by sending LC\_Init message containing a 64-bit pseudo-random nonce  $r_n$  to the HDCP Receiver.
- Computes 256-bit  $L = \text{HMAC-SHA256}(r_n \parallel r_n, k_d \text{ XOR } r_{rx})$  where HMAC-SHA256 is computed over  $r_n \parallel r_n$  and the key used for HMAC is  $k_d \text{ XOR } r_{rx}$  where  $r_{rx}$  is XOR-ed with the least significant 64-bits of  $k_d$ . All values are in big-endian order.
- On receiving the RTT\_Ready\_message from the receiver, the transmitter sends an RTT\_Challenge message containing the least significant 128-bits of  $L$ .
- Sets its watchdog to 7 ms. Locally check fails if the watchdog timer expires before LC\_Send\_L\_prime message is received.
- On receiving LC\_Send\_L\_prime message, the HDCP Transmitter compares the received value with the most significant 128-bits of  $L$  and locality check fails if there is a mismatch.

An HDCP Repeater initiates locality check on all its downstream HDCP-protected interface ports by sending unique  $r_n$  values to the connected HDCP Devices.

Figure 2.4 and 2.5 illustrates locality check between the HDCP Transmitter and HDCP Receiver.



**Figure 2.4. Locality Check between HDCP Transmitter and HDCP Receiver**



**Figure 2.5. Locality Check between HDCP Transmitter and HDCP Receiver (Pre-compute L and L')**

If the HDCP Transmitter is HDCP 2.0 compliant or if the `TRANSMITTER_LOCALITY_PRECOMPUTE_SUPPORT` bit received as part of the `AKE_Transmitter_info` message is set to zero or the receiver has set the `RECEIVER_LOCALITY_PRECOMPUTE_SUPPORT` bit to zero in its `AKE_Receiver_info` message, the HDCP Receiver

- Computes a 256-bit value  $L' = \text{HMAC-SHA256}(r_n, k_d \text{ XOR } r_{rx})$ .
- Sends `LC_Send_L_prime` message containing 256-bit  $L'$ .

If the `TRANSMITTER_LOCALITY_PRECOMPUTE_SUPPORT` bit received as part of the `AKE_Transmitter_info` message is set to one and the receiver has set the `RECEIVER_LOCALITY_PRECOMPUTE_SUPPORT` bit to one in its `AKE_Receiver_info` message, the HDCP Receiver

- Computes a 256-bit value  $L' = \text{HMAC-SHA256}(r_n, k_d \text{ XOR } r_{rx})$  if `AKE_Transmitter_info.VERSION` = `0x01`.
- Computes a 256-bit value  $L' = \text{HMAC-SHA256}(r_n || r_n, k_d \text{ XOR } r_{rx})$  if `AKE_Transmitter_info.VERSION` is not equal to `0x01`.
- Sends `RTT_Ready` message to the transmitter when  $L'$  calculation is complete and the receiver is ready for the RTT challenge
- On receiving the `RTT_Challenge` message from the transmitter, if the value received in the `RTT_Challenge` message matches the least significant 128 bits of  $L'$ , the receiver send the `LC_Send_L_prime` message containing the most significant 128-bits of  $L'$ .

In the case of a locality check failure (timeout or mismatch of  $L$  and  $L'$ ) between the HDCP Transmitter and HDCP Receiver, locality check must be reattempted by the HDCP Transmitter two additional times (for a total of three consecutive times) with the transmission of an LC\_Init message containing a new  $r_n$ . Three consecutive locality check failures results in an authentication failure and the authentication protocol is aborted (See Section **Error! Reference source not found.** on handling authentication failures).

## 2.4 Session Key Exchange

Successful completion of AKE and locality check stages affirms to HDCP Transmitter that the HDCP Receiver is authorized to receive HDCP Content. Session Key Exchange (SKE) is initiated by the HDCP Transmitter after a successful locality check. The HDCP Transmitter sends encrypted Session Key to the HDCP Receiver and enables HDCP Encryption 200 ms after sending encrypted Session Key. Content encrypted with the Session Key  $k_s$  starts to flow between the HDCP Transmitter and HDCP Receiver. HDCP Encryption must be enabled only after successful completion of AKE, locality check and SKE stages.

During SKE, the HDCP Transmitter

- Generates a pseudo-random 128-bit Session Key  $k_s$  and 64-bit pseudo-random number  $r_{iv}$ .
- Computes a 256-bit HMAC of  $r_{iv}$  – HMAC-SHA256( $r_{iv}$ ,  $k_d$ ) where HMAC-SHA256 is computed over  $r_{iv}$  and the key used for HMAC is  $k_d$ , if the HDCP receiver is compliant with HDCP 2.3 or higher (i.e. AKE\_Receiver\_Info.VERSION =0x03 or higher)
- Performs key derivation as explained in Section 2.7 to generate 128-bit  $dkey_2$  where  $dkey_2$  is the derived key when  $ctr=2$ .
- Computes 128-bit  $E_{dkey}(k_s) = k_s \text{ XOR } (dkey_2 \text{ XOR } r_{rx})$ , where  $r_{rx}$  is XORed with the least-significant 64-bits of  $dkey_2$ .
- Sends SKE\_Send\_Eks message containing  $E_{dkey}(k_s)$ ,  $r_{iv}$  and HMAC of  $r_{iv}$  to the HDCP Receiver.

On receiving SKE\_Send\_Eks message, the HDCP Receiver

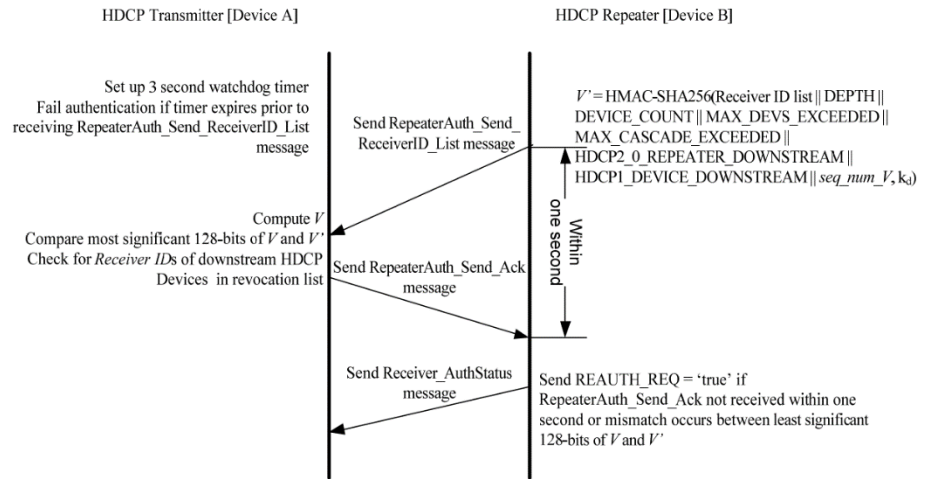
- Computes a 256-bit HMAC of  $r_{iv}$  – HMAC-SHA256( $r_{iv}$ ,  $k_d$ ) if the HDCP transmitter is compliant with HDCP 2.3 or higher (i.e. AKE\_Transmitter\_Info.VERSION =0x03 or higher). Compare the computed HMAC with HMAC that received as part of the SKE\_Send\_Eks message. A mismatch of the HMAC values results in an authentication failure and the authentication protocol is aborted.
- Performs key derivation as explained in Section 2.7 to generate 128-bit  $dkey_2$  where  $dkey_2$  is the derived key when  $ctr=2$ .
- Computes  $k_s = E_{dkey}(k_s) \text{ XOR } (dkey_2 \text{ XOR } r_{rx})$

## 2.5 Authentication with Repeaters

The HDCP Transmitter executes authentication with repeaters after Session Key exchange and only when REPEATER is 'true', indicating that the connected HDCP Receiver is an HDCP Repeater. Authentication with repeaters stage is used for the upstream propagation of topology information and the downstream propagation of Content Stream management information as explained in Section 2.5.1 and Section 2.5.2 respectively.

Authentication with repeaters may be implemented by the HDCP Transmitter in parallel with the flow of encrypted content and Link Synchronization. The Link Synchronization process is explained in Section 2.5.2.

### 2.5.1 Upstream Propagation of Topology Information



**Figure 2.6. Upstream Propagation of Topology Information**

Figure 2.6 illustrates upstream propagation of topology information. This stage assembles a list of all downstream *Receiver IDs* connected to the HDCP Repeater through a permitted connection tree, enabling revocation support upstream. This stage is implemented after successful completion of Session Key Exchange. This is used to assemble the latest topology information at the beginning of the HDCP Session immediately flowing an SKE or on subsequent changes to the topology due to connect or disconnect of an HDCP Receiver or HDCP Repeater.

HDCP Repeaters assemble the list of all connected downstream HDCP Receivers as the downstream HDCP-protected Interface Ports of the HDCP Repeater successfully complete the Authentication and Key Exchange and Locality check stages with connected HDCP Receivers. The list is represented by a contiguous set of bytes, with each *Receiver ID* occupying five bytes stored in big-endian order. The total length of the Receiver ID list is five bytes times the total number of connected and active downstream HDCP Devices, including downstream HDCP Repeaters, with which the HDCP Repeater has successfully completed the authentication protocol. This total number is represented in the RepeaterAuth\_Send\_ReceiverID list message by the DEVICE\_COUNT value. An HDCP-protected Interface Port with no active device connected adds nothing to the list. Also, the *Receiver ID* of the HDCP Repeater itself at any level is not included in its own Receiver ID list. An HDCP-protected Interface Port connected to an HDCP Receiver that is not an HDCP Repeater adds the *Receiver ID* of the connected HDCP Receiver to the list. HDCP-protected Interface Ports that have an HDCP Repeater connected add the Receiver ID list received from the connected downstream HDCP Repeater, plus the *Receiver ID* of the connected downstream HDCP Repeater itself.

In order to add the Receiver ID list of the connected HDCP Repeater, it is necessary for the HDCP Repeater to verify the integrity of the list. If the connected HDCP Repeater is not an HDCP 2.0-compliant Device, the HDCP Repeater verifies the integrity by computing  $V$  and checking the most

significant 128-bits of  $V$  against most significant 128 bits of  $V'$  received as part of the RepeaterAuth\_Send\_ReceiverID\_List message from the connected downstream HDCP Repeater. If the connected HDCP Repeater is an HDCP 2.0-compliant Device, the HDCP Repeater verifies the integrity of the list by computing  $V$  and comparing  $V$  against  $V'$ . If the values do not match, the downstream Receiver ID list integrity check fails, and the HDCP Repeater must not add the Receiver ID list received from the downstream HDCP repeater to its Receiver ID list.

When the HDCP Repeater has assembled the complete list of *Receiver IDs* of connected and active HDCP Devices with which the HDCP Repeater has successfully completed the authentication protocol, it computes the 256-bit verification value  $V'$ .

An HDCP Repeater connected to an HDCP 2.0-compliant version upstream HDCP Transmitter and HDCP Transmitter connected to an HDCP 2.0-compliant HDCP Repeater computes respective  $V$  and  $V'$  values as given below. HMAC-SHA256 is computed over the concatenation of Receiver ID list, DEPTH, DEVICE\_COUNT, MAX\_DEVS\_EXCEEDED and MAX\_CASCADE\_EXCEEDED received as part of the RepeaterAuth\_Send\_ReceiverID\_List message. The key used for HMAC is  $k_d$ .

$$V' \text{ (or } V) = \text{HMAC-SHA256}(\text{Receiver ID list} \parallel \text{DEPTH} \parallel \text{DEVICE\_COUNT} \parallel \text{MAX\_DEVS\_EXCEEDED} \parallel \text{MAX\_CASCADE\_EXCEEDED}, k_d)$$

An HDCP Repeater connected to an upstream HDCP Transmitter that is not 2.0-compliant version upstream HDCP Transmitter and HDCP Transmitter connected to an HDCP Repeater that is not an HDCP 2.0-compliant computes respective  $V$  and  $V'$  values as given below. HMAC-SHA256 is computed over the concatenation of Receiver ID list, DEPTH, DEVICE\_COUNT, MAX\_DEVS\_EXCEEDED, MAX\_CASCADE\_EXCEEDED, HDCP2\_LEGACY\_DEVICE\_DOWNSTREAM, HDCP1\_DEVICE\_DOWNSTREAM and  $seq\_num\_V$  received as part of the RepeaterAuth\_Send\_ReceiverID\_List message. The key used for HMAC is  $k_d$ .

$$V' \text{ (or } V) = \text{HMAC-SHA256}(\text{Receiver ID list} \parallel \text{DEPTH} \parallel \text{DEVICE\_COUNT} \parallel \text{MAX\_DEVS\_EXCEEDED} \parallel \text{MAX\_CASCADE\_EXCEEDED} \parallel \text{HDCP2\_LEGACY\_DEVICE\_DOWNSTREAM} \parallel \text{HDCP1\_DEVICE\_DOWNSTREAM} \parallel \text{seq\_num\_V}, k_d)$$

Receiver ID list is formed by appending downstream *Receiver IDs* in big-endian order. When the Receiver ID list, DEPTH, DEVICE\_COUNT, and if applicable HDCP2\_LEGACY\_DEVICE\_DOWNSTREAM and HDCP1\_DEVICE\_DOWNSTREAM are available, the HDCP Repeater sends RepeaterAuth\_Send\_ReceiverID\_List message to the upstream HDCP Transmitter. The HDCP Repeater sends  $V'$  if the upstream transmitter is HDCP 2.0-compliant and the most significant 128-bits of  $V'$  if the upstream transmitter is not HDCP 2.0-compliant.

The HDCP Repeater initializes  $seq\_num\_V$  to 0 at the beginning of HDCP session i.e. after  $r_{tx}$  is received. It is incremented by one after the transmission of every RepeaterAuth\_Send\_ReceiverID\_List message.  $seq\_num\_V$  must be never reused during an HDCP Session for the computation of  $V$  (or  $V'$ ). If  $seq\_num\_V$  rolls over, the HDCP Transmitter must detect the roll-over in the RepeaterAuth\_Send\_ReceiverID\_List receiver from the HDCP Repeater and the transmitter must disable HDCP Encryption if encryption is enabled, restart authentication by the transmission of a new  $r_{tx}$  as part of the AKE\_Init message.

When an HDCP Repeater receives HDCP2\_LEGACY\_DEVICE\_DOWNSTREAM = 'true' or HDCP1\_DEVICE\_DOWNSTREAM = 'true' from a downstream HDCP Repeater, it must

propagate the information to the upstream HDCP Transmitter by setting the corresponding values to 'true' in the RepeaterAuth\_Send\_ReceiverID\_List message.

If `HDCP2_LEGACY_DEVICE_DOWNSTREAM` = 'true' or `HDCP1_DEVICE_DOWNSTREAM` = 'true', the Upstream Content Control Function may instruct the most upstream HDCP Transmitter to abort the transmission of certain HDCP encrypted type 1 Content Streams. The most upstream HDCP Transmitter must be prepared to process the request and immediately cease the transmission of specific Content Streams as instructed by the Upstream Content Control Function.

Whenever RepeaterAuth\_Send\_ReceiverID\_List message is received, the HDCP Transmitter verifies the integrity of the Receiver ID list by computing  $V$  and comparing either  $V$  and  $V'$  (if connected HDCP Repeater is HDCP 2.0-compliant) or most significant 128-bits of  $V$  and  $V'$  (if connected HDCP Repeater is not HDCP 2.0-compliant). If the values do not match, authentication fails and, the authentication protocol is aborted and HDCP encryption is disabled.

On successful verification of Receiver ID list and topology information, i.e. if the values match, none of the reported Receiver IDs are in the current revocation list (in the case of most upstream HDCP Transmitter), the HDCP Transmitter does not detect a roll-over of  $seq\_num\_V$ , the downstream topology does not exceed specified maximums (explained below) and the HDCP Repeater is not HDCP 2.0 compliant, the HDCP Transmitter (including downstream port of HDCP Repeater) sends the least significant 128-bits of  $V$  to the HDCP Repeater as part of RepeaterAuth\_Send\_Ack message. Every RepeaterAuth\_Send\_ReceiverID\_List message from the transmitter to repeater on successful verification of Receiver ID list and topology information by the transmitter.

The RepeaterAuth\_Send\_Ack message must be received by the HDCP Repeater within one second from the transmission of the RepeaterAuth\_Send\_ReceiverID\_List message to the HDCP Transmitter if the HDCP Transmitter is not HDCP 2.0-compliant and the downstream topology does not exceed specified maximums. A match between the least significant 128-bits of  $V$  and  $V'$  indicates successful upstream transmission of topology information. If a mismatch occurs or RepeaterAuth\_Send\_Ack message is not received within one second, the HDCP Repeater must send the RepeaterAuth\_Send\_Ack message with the REAUTH\_REQ set to 'true' and must transition in to the unauthenticated state (see Section 2.10.3).

If the upstream HDCP Transmitter receives a Receiver\_AuthStatus message with REAUTH\_REQ set to 'true', it may initiate re-authentication with the HDCP Repeater by the transmission of a new  $r_x$ .

After transmitting the SKE\_Send\_Eks message, the HDCP Transmitter, having determined that REPEATER received earlier in the protocol is 'true', sets a three second watchdog timer. If the RepeaterAuth\_Send\_ReceiverID\_List message is not received by the HDCP Transmitter within a maximum –permitted time of three seconds after transmitting SKE\_Send\_Eks message, authentication of the HDCP Repeater fails. With this failure, the HDCP Transmitter disables HDCP Encryption and aborts the authentication protocol with the HDCP Repeater.

When an HDCP Receiver (including HDCP Repeater) is connected to the HDCP Repeater or when a connected, active HDCP Receiver with which the HDCP Repeater has successfully completed the authentication protocol is disconnected from the HDCP Repeater and the upstream HDCP Transmitter is not HDCP 2.0-compliant, the HDCP Repeater must send the

RepeaterAuth\_Send\_ReceiverID\_List message to the upstream HDCP Transmitter which must include the Receiver IDs of all connected and active downstream HDCP Receivers with which the

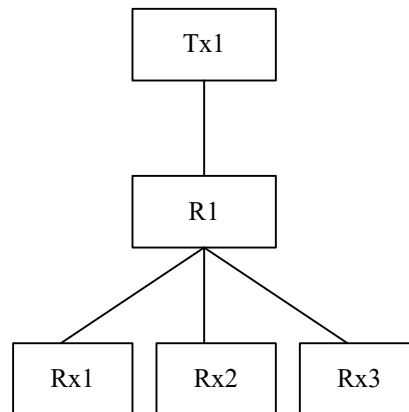


HDCP Repeater has successfully completed the authentication protocol. This enables upstream propagation of the most recent topology information after changes to the topology without interrupting the transmission of HDCP Content.

Refer to Table 2.2 for the HDCP Repeater upstream and downstream propagation time.

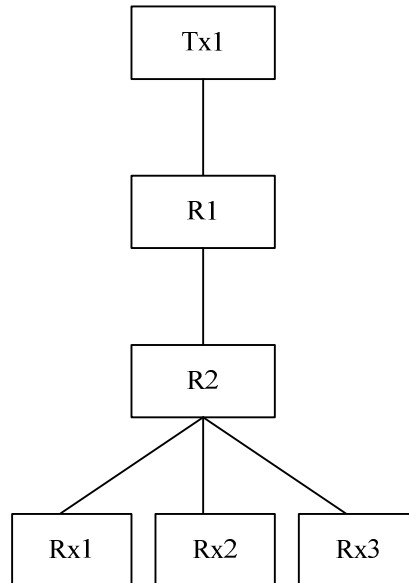
The HDCP Repeater propagates topology information upward through the connection tree to the HDCP Transmitter. An HDCP Repeater reports the topology status variables `DEVICE_COUNT` and `DEPTH`. The `DEVICE_COUNT` for an HDCP Repeater is equal to the total number of connected downstream HDCP Receivers and HDCP Repeaters. The value is calculated as the sum of the number of directly connected downstream HDCP Receivers and HDCP Repeaters plus the sum of the `DEVICE_COUNT` received from all connected HDCP Repeaters. The `DEPTH` status for an HDCP Repeater is equal to the maximum number of connection levels below any of the downstream HDCP-protected Interface Ports. The value is calculated as the maximum `DEPTH` reported from downstream HDCP Repeaters plus one (accounting for the connected downstream HDCP Repeater).

In Figure 2.7, R1 has three downstream HDCP Receivers connected to it. It reports a `DEPTH` of one and a `DEVICE_COUNT` of three.



**Figure 2.7. DEPTH and DEVICE\_COUNT for HDCP Repeater**

In Figure 2.8, R1 reports a `DEPTH` of two and a `DEVICE_COUNT` of four.



**Figure 2.8. DEPTH and DEVICE\_COUNT for HDCP Repeater**

HDCP Repeaters must be capable of supporting `DEVICE_COUNT` values less than or equal to 31 and `DEPTH` values less than or equal to 4. If the computed `DEVICE_COUNT` for an HDCP Repeater exceeds 31, the error is referred to as `MAX_DEVS_EXCEEDED` error. The repeater sets `MAX_DEVS_EXCEEDED = 'true'` in the `RepeaterAuth_Send_ReceiverID_List` message. If the computed `DEPTH` for an HDCP Repeater exceeds four, the error is referred to as `MAX_CASCADE_EXCEEDED` error. The repeater sets `MAX_CASCADE_EXCEEDED = 'true'` in the `RepeaterAuth_Send_ReceiverID_List` message. When an HDCP Repeater receives a `MAX_DEVS_EXCEEDED` or a `MAX_CASCADE_EXCEEDED` error from a downstream HDCP Repeater, it must propagate the error to the upstream HDCP Transmitter and must not transmit  $V'$  (or the most significant 128-bits of  $V'$ ), `DEPTH`, `DEVICE_COUNT`, and Receiver ID list and if applicable `HDCP2_LEGACY_DEVICE_DOWNSTREAM` and `HDCP1_DEVICE_DOWNSTREAM`.

Authentication fails if the topology maximums are exceeded. HDCP Encryption is disabled and the authentication protocol is aborted. The top-level HDCP Transmitter, having already performed SRM integrity check during AKE, proceeds to see if the *Receiver ID* of any downstream device is found in the current revocation list, and, if present, authentication fails, HDCP Encryption is disabled and authentication protocol is aborted.

In some instances, certain Upstream Content Protocol Functions may implement functionality to perform revocation checking of the downstream HDCP Receivers (including HDCP Repeaters). In such instances, and if requested by Upstream Content Protocol Function, the Top-Level HDCP Transmitter must pass the downstream Receiver IDs, including the receiver ID of the attached downstream HDCP Receiver or HDCP Repeater and any Receiver IDs received as part of the Receiver ID list, to such Upstream Content Protocol Function. If Top-Level HDCP Transmitter receives an indication from the Upstream Content Protocol Function that a downstream device has been found to be revoked, the Top-Level HDCP Transmitter must fail authentication, disable HDCP encryption and abort the authentication protocol.



Figure 2.9. HDCP Repeater Protocol Timing Requirements

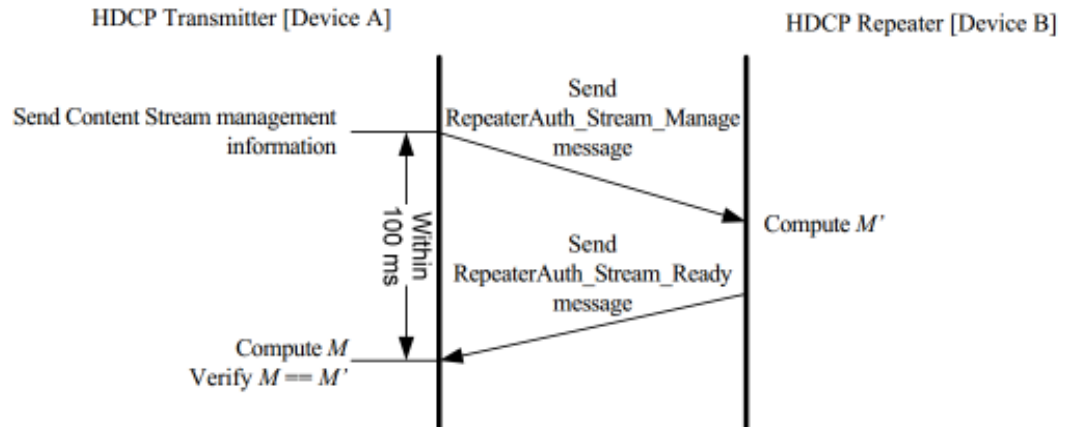
From	To	Max Delay	Conditions and Comments
SKE_Send_Eks1 Session Key received from	SKE_Send_Eks2 $k_s$ generated by HDCP Repeater	100 ms	Downstream propagation time.

Upstream HDCP Transmitter	transmitted downstream		
SKE_Send_Eks3 $k_s$ transmitted to all downstream HDCP-protected Interface Ports	RepeaterAuth_Send_ReceiverID_List1 <i>Receiver IDs</i> and topology information transmitted upstream	200 ms	Upstream propagation time when no downstream HDCP Repeaters are attached (no downstream Receiver ID lists to process)
RepeaterAuth_Send_ReceiverID_List1 Downstream <i>Receiver IDs</i> and topology information received	RepeaterAuth_Send_ReceiverID_List2 <i>Receiver IDs</i> and topology information transmitted upstream	200 ms	Upstream propagation time when one or more HDCP Repeaters are attached. From latest downstream RepeaterAuth_Send_ReceiverID_List message. (downstream Receiver ID lists must be processed)
SKE_Send_Eks1 Upstream HDCP Transmitter transmits $k_s$	RepeaterAuth_Send_ReceiverID_List2 Upstream HDCP Transmitter receives RepeaterAuth_Send_ReceiverID_List message	1.2 seconds	For the Maximum of four repeater levels, 4 * (100 ms + 200 ms)

Table 2.2. HDCP Repeater Protocol Timing Requirements

Table 2.2 specifies HDCP Repeater timing requirements that bound the worst-case propagation time for the Receiver ID list. A maximum delay of three seconds has been provided, to receive the RepeaterAuth\_Send ReceiverID\_List message by the upstream transmitter, to account for authentication delays due to the presence of downstream receivers that have not been paired with the upstream HDCP Repeater. Note that because each HDCP Repeater does not know the number of downstream HDCP Repeaters, it must use the same three-second timeout used by the upstream HDCP Transmitter for receiving the RepeaterAuth\_Send ReceiverID\_List message.

## 2.5.2 Downstream Propagation of Content Stream Management Information



**Figure 2.10. Downstream Propagation of Content Stream Management Information**

The HDCP Transmitter may transmit multiple Content Streams to an HDCP Receiver during an HDCP Session. The HDCP Transmitter may use the same Session Key,  $ks$ , negotiated during the HDCP Session for HDCP Encryption of the Content Streams.

The HDCP Transmitter propagates Content Stream management information, which includes Type values assigned to Content Streams, using the RepeaterAuth\_Stream\_Manage message to the attached HDCP Repeater only if the attached HDCP Repeater is not an HDCP 2.0-compliant Device. The HDCP Transmitter executes this step after successful completion of Session Key Exchange and before beginning the transmission of a Content Stream after HDCP Encryption to the HDCP Repeater. The RepeaterAuth\_Stream\_Manage message from an HDCP Transmitter to the attached HDCP Repeater identifies restrictions, as specified by the Upstream Content Control Function, on the transmission of Content Streams to specific devices.

Type values are assigned to all Content Streams by the most upstream HDCP Transmitter based on instructions received from the Upstream Content Control Function. The exact mechanism used by the Upstream Content Control Function to instruct the HDCP Transmitter is outside the scope of this specification. Type 0 Content Streams (see Section 4.3.15) may be transmitted by the HDCP Repeater to all HDCP Devices. Type 1 Content Streams (see Section 4.3.15) must not be transmitted by the HDCP Repeater through its HDCP-protected Interface Ports connected to HDCP 1.x-compliant Devices, HDCP 2.0-compliant Repeaters, and HDCP 2.1-compliant Devices.

The most upstream HDCP Transmitter must not transmit Type 1 Content Streams to HDCP 1.x-compliant Devices, HDCP 2.0-compliant Repeaters and HDCP 2.1-compliant Devices as instructed by the corresponding Upstream Content Control Function.

The HDCP Transmitter must send the RepeaterAuth\_Stream\_Manage message specifying Type values assigned to Content Streams, to the attached HDCP Repeater at least 100ms before the transmission of the corresponding Content Streams after HDCP Encryption. The HDCP Transmitter must only send the RepeaterAuth\_Stream\_Manage message corresponding to encrypted Content Streams it will transmit to the HDCP Repeater. The HDCP Transmitter initializes  $seq\_num\_M$  to 0

at the beginning of the HDCP Session i.e. after rtx, is sent. It is incremented by one after the transmission of every RepeaterAuth\_Stream\_Manage message.

On receiving the RepeaterAuth\_Stream\_Manage message, the HDCP Repeater computes M' as given below. HMAC-SHA256 is computed over the concatenation of STREAMID\_TYPE (see Section 4.3.15) and seq\_num\_M values received as part of the RepeaterAuth\_Stream\_Manage message. All values are in big-endian order. The key used for HMAC is SHA256 (kd).

seq\_num\_M must never be reused during an HDCP Session for the computation of M' (or M). If seq\_num\_M rolls over, the HDCP Transmitter must disable HDCP Encryption if encryption is enabled, restart authentication by the transmission of a new rtx as part of the AKE\_Init message.

$M' \text{ (or } M) = \text{HMAC-SHA256}(\text{STREAMID\_TYPE} \parallel \text{seq\_num\_M}, \text{SHA256 (kd)})$ .

M' must be sent by the HDCP Repeater to the HDCP Transmitter as part of the RepeaterAuth\_Stream\_Ready message.

The HDCP Transmitter must receive the RepeaterAuth\_Stream\_Ready message within 100 ms after the transmission of RepeaterAuth\_Stream\_Manage message. Every RepeaterAuth\_Stream\_Manage message from the transmitter to the repeater must be followed by a RepeaterAuth\_Stream\_Ready message from the repeater to the transmitter.

When the RepeaterAuth\_Stream\_Ready message is received, the HDCP Transmitter verifies the integrity of the message by computing M and comparing this value to M'. If M is equal to M', the HDCP Transmitter may transmit the Content Streams identified in the corresponding RepeaterAuth\_Stream\_Manage message. If the RepeaterAuth\_Stream\_Ready message is not received within 100 ms or if M is not equal to M' the HDCP Transmitter must not transmit the Content Streams identified in the corresponding RepeaterAuth\_Stream\_Manage message. Type value is assigned to each Content Stream through the successful transmission/reception of a single RepeaterAuth\_Stream\_Manage message. The Content Stream shall be associated with such type value throughout the HDCP session.

An HDCP Repeater connected to an HDCP 2.0-compliant Transmitter or an HDCP 1.x-compliant Transmitter will not receive the RepeaterAuth\_Stream\_Manage message from the transmitter. In this case, the HDCP Repeater must assign a Type value of 0x00 to all Content Streams received from the HDCP Transmitter.

The HDCP Repeater must in turn propagate the received Content Stream management information using the RepeaterAuth\_Stream\_Manage message further downstream.

## 2.6 Link Synchronization

After successful completion of SKE, HDCP Encryption is enabled and encrypted content starts to flow between the HDCP Transmitter and the HDCP Receiver. As explained in Section 3.4, the presence of the *coarseInputCtr* field in the WHDI frame Extended Header indicates that HDCP Encryption is enabled and the payload is encrypted. Once encrypted content starts to flow, a periodic Link Synchronization is performed to maintain cipher synchronization between the HDCP Transmitter and the HDCP Receiver.

Link Synchronization is achieved every time a frame Extended Header is transmitted, by the inclusion of *coarseInputCtr* in the header. (See Section 3.3 for details about *coarseInputCtr*). The HDCP Receiver updates its *coarseInputCtr* with the corresponding counter values received from the transmitter.

## 2.7 Key Derivation

Key derivation is illustrated in Figure 2..

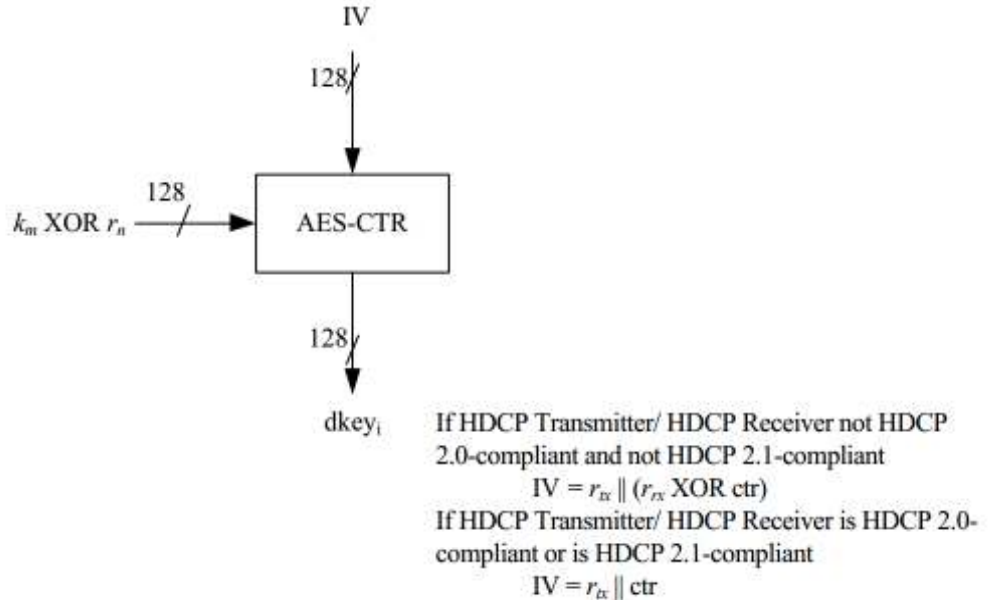


Figure 2.11. Key Derivation

The HDPC transmitter sets  $IV = r_{tx} \parallel (r_{tx} \text{ XOR } ctr)$  if the HDPC receiver is not 2.0 compliant and is not HDPC 2.1 compliant (i.e. `AKE_Receiver_Info.VERSION` is not equal to 0x01), if the HDPC receiver is 2.0 compliant or HDPC 2.1 compliant (i.e. `AKE_Receiver_Info.VERSION` is equal to 0x01), HDPC transmitter sets  $IV = r_{tx} \parallel ctr$ . All values are in big-endian order.

The HDPC receiver sets  $IV = r_{tx} \parallel (r_{tx} \text{ XOR } ctr)$  if the HDPC transmitter is not 2.0 compliant and is not HDPC 2.1 compliant (i.e. `AKE_Transmitter_Info.VERSION` is not equal to 0x01), if the HDPC transmitter is 2.0 compliant or HDPC 2.1 compliant (i.e. `AKE_Transmitter_Info.VERSION` is equal to 0x01), HDPC receiver sets  $IV = r_{tx} \parallel ctr$ . All values are in big-endian order.

$r_{tx}$  and  $ctr$  are in big-endian order.  $ctr$  is a 64-bit counter and is initialized to 0 at the beginning of the HDPC Session i.e. after  $r_{tx}$  is sent or received. It is incremented by one after every derived key computation.  $dkey_i$  is the 128-bit derived key when  $ctr = i$ .  $ctr$  must never be reused during an HDPC Session.

$r_n$  is initialized to 0 during AKE i.e. during the generation of  $dkey_0$  and  $dkey_1$ . It is set to a pseudo-random value during locality check as explained in Section 2.3. The pseudo-random  $r_n$  is XORed with the least-significant 64-bits of  $k_m$  during generation of  $dkey_2$ .

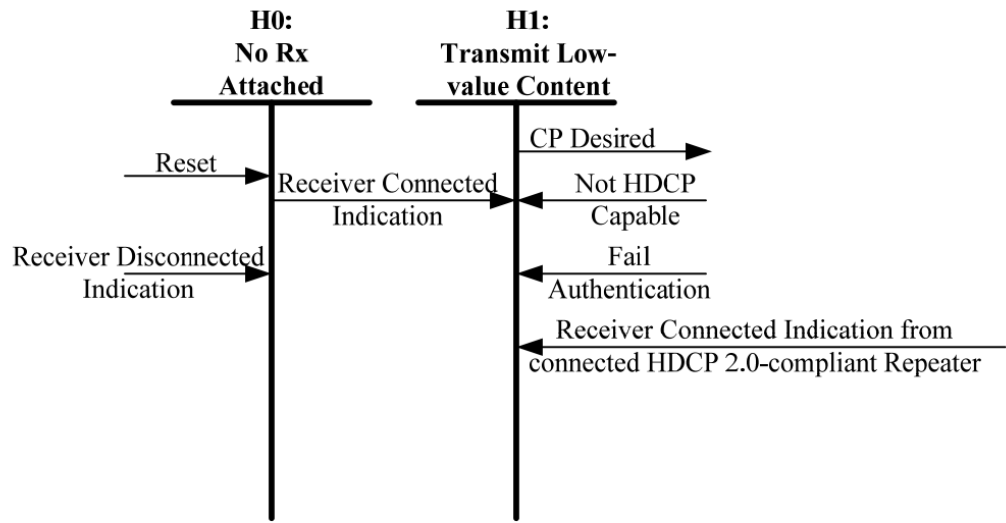
## 2.8 HDPC Transmitter State Diagram

As explained in Section 1.3, the HDPC Transmitter may support simultaneous connections to HDPC Receivers through one or more of its HDPC-protected interface ports. The HDPC Transmitter state diagram is implemented independently on each HDPC-protected interface port.

The HDCP Transmitter Link State Diagram and HDCP Transmitter Authentication Protocol State Diagram (Figure 2. and Figure 2.) illustrate the operation states of the authentication protocol for an HDCP Transmitter that is not an HDCP Repeater. For HDCP Repeaters, the downstream (HDCP Transmitter) side is covered in Section 2.10.2.

Transmitter’s decision to begin authentication is dependent on events such as detection of an HDCP Receiver, availability of premium content or other implementation dependent details in the transmitter. In the event of the authentication failure, an HDCP Receiver must be prepared to process subsequent authentication attempts. The HDCP Transmitter may cease to attempt authentication for transmitter-specific reasons, which include receiving a Receiver Disconnected Indication or after a certain number of authentication re-attempts by the transmitter.

The transmitter must not initiate authentication unless it determines that the receiver is HDCP-capable. The method used by the HDCP Transmitter to determine whether the receiver is HDCP-capable is outside the scope of this specification.



Note: Transition arrows with no connected state (e.g. Reset) indicate transitions that can occur from multiple states

**Figure 2.12. HDCP Transmitter Link State Diagram**

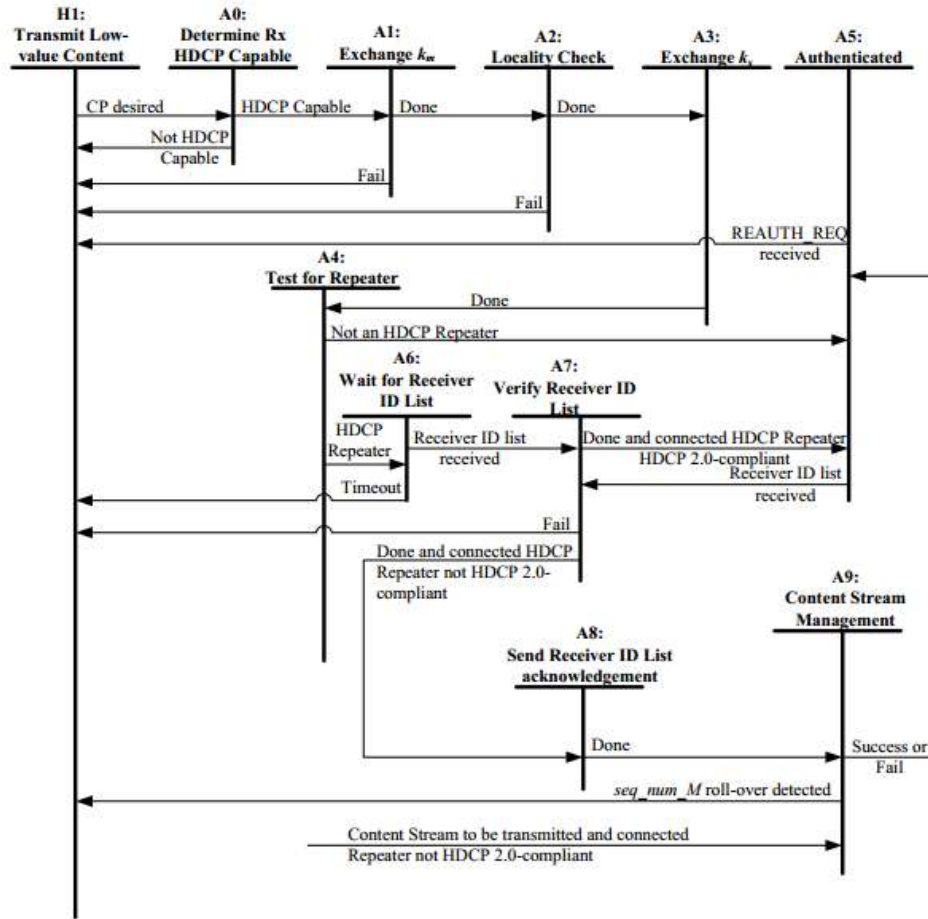


Figure 2.9. HDCP Transmitter Authentication Protocol State Diagram

**Transition Any State:H0.** Reset conditions at the HDCP Transmitter or disconnect of the connected HDCP capable receiver cause the HDCP Transmitter to enter the No Receiver Attached state.

**Transition H0:H1.** The detection of a sink device (through Receiver Connected Indication) indicates to the transmitter that a sink device is connected and ready to display the received content. When the receiver is no longer active, the transmitter is notified through Receiver Disconnected Indication.

**State H1: Transmit Low-value Content.**

In this state the transmitter may begin sending an unencrypted signal with HDCP Encryption disabled. The transmitted signal can be a low value content or informative on-screen display. If low-value content is transmitted, this will ensure that a valid video signal is displayed to the user before and during authentication. At any time a Receiver Connected Indication received from the connected HDCP 2.0-compliant HDCP Repeater causes the transmitter to transition in to this state.

**Transition H1:A0.** If content protection is desired by the Upstream Content Control Function, then the HDCP Transmitter should immediately attempt to determine whether the receiver is HDCP capable.

**State A0: Determine Rx HDCP Capable.** The transmitter determines that the receiver is HDCP capable as part of the setup and discovery procedures defined in the AVCL chapter of the WHDI



specification. Since state A0 is reached when content protection is desired by the Upstream Content Control Function, authentication must be started immediately by the transmitter if the receiver is HDCP capable. A valid video screen is displayed to the user with encryption disabled during this time.

**Transition A0:H1.** If the receiver is not HDCP capable, the transmitter may continue to transmit low value content or informative on-screen display.

**Transition A0:A1.** If the receiver is HDCP capable, the transmitter initiates the authentication protocol.

**State A1: Exchange  $k_m$ .**

In this state, the HDCP Transmitter initiates authentication by sending AKE\_Init message containing  $r_{tx}$  to the HDCP Receiver. It receives AKE\_Send\_Cert from the receiver containing REPEATER and  $cert_{rx}$  and AKE\_Receiver\_Info message (if the HDCP Receiver is not HDCP 2.0-compliant). If the HDCP Transmitter does not receive AKE\_Receiver\_Info message within 100 ms of the transmission of AKE\_Transmitter\_Info message, it indicates that the HDCP Receiver is an HDCP 2.0-compliant Device.

If the HDCP Transmitter does not have  $k_m$  stored corresponding to the Receiver ID, it generates  $E_{k_{pub}}(k_m)$  and sends  $E_{k_{pub}}(k_m)$  as part of the AKE\_No\_Stored\_km message to the receiver after verification of signature on  $cert_{rx}$ . It performs integrity check on the SRM and checks to see whether the Receiver ID of the connected HDCP Device is in the revocation list. It receives AKE\_Send\_rrx message containing  $r_{rx}$  from the receiver. It computes H, receives AKE\_Send\_H\_prime message from the receiver containing  $H'$  within one second after sending AKE\_No\_Stored\_km to the receiver and compares  $H'$  against H.

If the HDCP Transmitter has  $k_m$  stored corresponding to the Receiver ID, it sends AKE\_Stored\_km message containing  $E_{k_h}(k_m)$  and  $m$  to the receiver, performs integrity check on the SRM, checks to see whether the Receiver ID of the connected HDCP Device is in the revocation list and receives  $r_{rx}$  as part of AKE\_Send\_rrx message from the receiver. It computes H, receives AKE\_Send\_H\_prime message from the receiver containing  $H'$  within 200 ms after sending AKE\_Stored\_km to the receiver and compares  $H'$  against H.

If the HDCP Transmitter does not have a  $k_m$  stored corresponding to the Receiver ID, it implements pairing with the HDCP Receiver as explained in Section 2.2.1.

**Transition A1:H1.** This transition occurs on failure of signature verification on  $cert_{rx}$ , failure of SRM integrity check, if Receiver ID of the connected HDCP Device is in the revocation list or if there is a mismatch between H and  $H'$ . This transition also occurs if AKE\_Send\_H\_prime message is not received within one second after sending AKE\_No\_Stored\_km or within 200 ms after sending AKE\_Stored\_km to the receiver.

**Transition A1:A2.** The HDCP Transmitter implements locality check after successful completion of AKE and pairing.

**State A2: Locality Check.** In this state, the HDCP Transmitter initiates locality check by sending LC\_Init message containing  $r_n$  to the HDCP Receiver, sets its watchdog timer to 7 ms and computes L. On receiving LC\_Send\_L\_prime message from the receiver, it compares  $L'$  against L.

**Transition A2:H1.** This transition occurs on three consecutive locality check failures. Locality check fails when  $L'$  is not received within 7 ms and the watchdog timer at the HDCP Transmitter expires or on a mismatch between L and  $L'$ .

**Transition A2:A3.** The HDCP Transmitter implements SKE after successful completion of locality check.

**State A3: Exchange  $k_s$ .** The HDCP Transmitter sends encrypted Session Key,  $E_{dk_{key}}(k_s)$ , and  $r_{iv}$  to the HDCP Receiver as part of the SKE\_Send\_Eks message. It enables HDCP Encryption 200 ms after sending encrypted Session Key. HDCP Encryption must be enabled only after successful completion of AKE, locality check and SKE stages.

**Transition A3:A4.** This transition occurs after completion of SKE.

**State A4: Test for Repeater.** The HDCP Transmitter evaluates the REPEATER value that was received in State A1.

**Transition A4:A5.** REPEATER is 'false' (the HDCP Receiver is not an HDCP Repeater).

**State A5: Authenticated.** At this time, and at no prior time, the HDCP Transmitter has completed the authentication protocol.

A periodic Link Synchronization is performed to maintain cipher synchronization between the HDCP Transmitter and the HDCP Receiver.

**Transition A4:A6.** REPEATER is 'true' (the HDCP Receiver is an HDCP Repeater).

**State A6: Wait for Receiver ID List.** The HDCP Transmitter sets up a three-second watchdog timer after sending SKE\_Send\_Eks.

**Transition A6:H1.** The watchdog timer expires before the RepeaterAuth\_Send\_ReceiverID\_List is received.

**Transition A6:A7.** RepeaterAuth\_Send\_ReceiverID\_List message is received.

**State A7: Verify Receiver ID List.**

Verify Receiver ID List. If a transition in to this state occurs from State A6, the watchdog timer is cleared. If both MAX\_DEVS\_EXCEEDED and MAX\_CASCADE\_EXCEEDED are not 'true', computes  $V$ . If the connected HDCP Repeater is HDCP 2.0-compliant, compares  $V$  and  $V'$ . If the connected HDCP Repeater is not HDCP 2.0-compliant, compares the most significant 128-bits of  $V$  and  $V'$ . The Receiver IDs from the Receiver ID list are compared against the current revocation list.

**Transition A7:H1.**

This transition is made if a mismatch occurs between  $V$  and  $V'$  (if the connected HDCP Repeater is HDCP 2.0-compliant) or the most significant 128-bits of  $V$  and  $V'$  (if the connected HDCP Repeater is not HDCP 2.0-compliant). This transition is also made if any of the Receiver IDs in the Receiver ID list are found in the current revocation list or if the HDCP Transmitter detects a roll-over of  $seq\_num$   $V$  (if the repeater is not HDCP 2.0-compliant). A MAX\_CASCADE\_EXCEEDED or MAX\_DEVS\_EXCEEDED error also causes this transition.

**Transition A7:A5.**

This transition occurs if the connected HDCP Repeater is HDCP 2.0-compliant, on successful verification of  $V$  and  $V'$ , none of the reported Receiver IDs are in the current revocation list, and the downstream topology does not exceed specified maximums.

**Transition A7:A8:** This transition occurs if the connected HDCP repeater is not HDCP 2.0-compliant, on successful verification of the most significant 128-bits of  $V$  and  $V'$ , none of the reported Receiver IDs are in the current revocation list, the HDCP Transmitter does not detect a roll-over of  $seq\_num\_V$  and the downstream topology does not exceed specified maximums.

**State A8:** Send Receiver ID list acknowledgement. The HDCP transmitter send the least significant 128-bits of  $V$  to the HDCP Repeater as part of the RepeaterAuth\_Send\_Ack message.

The RepeaterAuth\_Send\_Ack message must be received by the HDCP Repeater within one second

**Transition A8:A9.** This transition occurs after RepeaterAuth\_Send\_Ack message has been sent to repeater

**Transition A5:H1.** This transition occurs if a Reciver\_AuthStatus message with the REAUTH\_REQ set to true is received.

**Transition A5:A7.** This transition occurs whenever a RepeaterAuth\_Send\_RecieverID\_list message is received from the connected HDCP Repeater that is not HDCP 2.0 compliant.

#### **State A9:**

Content Stream Management. This stage is implemented if Content Stream is to be transmitted and the connected HDCP Repeater is not HDCP 2.0-compliant. The HDCP Transmitter sends the RepeaterAuth\_Stream\_Manage message specifying Type values assigned to Content Streams, to the attached HDCP Repeater at least 100ms before the transmission of the corresponding Content Streams after HDCP Encryption. It must receive the RepeaterAuth\_Stream\_Ready message from the HDCP Repeater within 100 ms after the transmission of RepeaterAuth\_Stream\_Manage message and verifies  $M'$ . This step fails if the RepeaterAuth\_Stream\_Ready message is not received within 100 ms or if  $M$  is not equal to  $M'$ .

This stage may be implemented in parallel with the upstream propagation of topology information (State A4, State A6, State A7 and State A8) and with the flow of encrypted content and Link Synchronization (State A5). This state may be implemented asynchronously from the rest of the state diagram. A transition in to this state may occur from State A4, State A5, State A6, State A7 or State A8 if Content Stream is to be transmitted and the connected HDCP Repeater is not HDCP 2.0-compliant. Also, the transition from State A9 must return to the appropriate state to allow for uninterrupted operation.

Note: The HDCP Transmitter must not transmit Type 1 Content Streams to HDCP 1.x-compliant Devices and HDCP 2.0-compliant Repeaters as instructed by the corresponding Upstream Content Control Function.

**Transition A9:A5.** This transition occurs on success of the Content Stream management stage.

**Transition A9:H1.** This transition occurs if  $seq\_num\_M$  rolls over.

Note: Since authentication with repeaters is implemented in parallel with the flow of encrypted content and Link Synchronization, the link synchronization process (i.e. State A5) must be implemented asynchronously from the rest of the state diagram. The transition into State A5 must occur from any state for which encryption is currently enabled. Also, the transition from state A5 returns to the appropriate state to allow for uninterrupted operation.

The HDCP Transmitter may support simultaneous connections to HDCP Receivers through one or more of its HDCP-protected interface ports. It may share the same Session Key and  $r_{iv}$  across all its

HDCP-protected interface ports, as explained in Section 0. However, the HDCP Transmitter must ensure that each connected HDCP Receiver receives distinct  $k_m$  and  $r_{tx}$  values.

## 2.9 HDCP Receiver State Diagram

The operation states of the authentication protocol for an HDCP Receiver that is not an HDCP Repeater are illustrated in Figure 2.10. For HDCP Repeaters, the upstream (HDCP Receiver) side is covered in Section 2.10.3.

The HDCP Receiver must be ready to re-authenticate with the HDCP Transmitter at any point in time. In particular, the only indication to the HDCP Receiver of a re-authentication attempt by the HDCP Transmitter is the reception of an  $r_{tx}$  as part of the AKE\_Init message from the HDCP Transmitter.

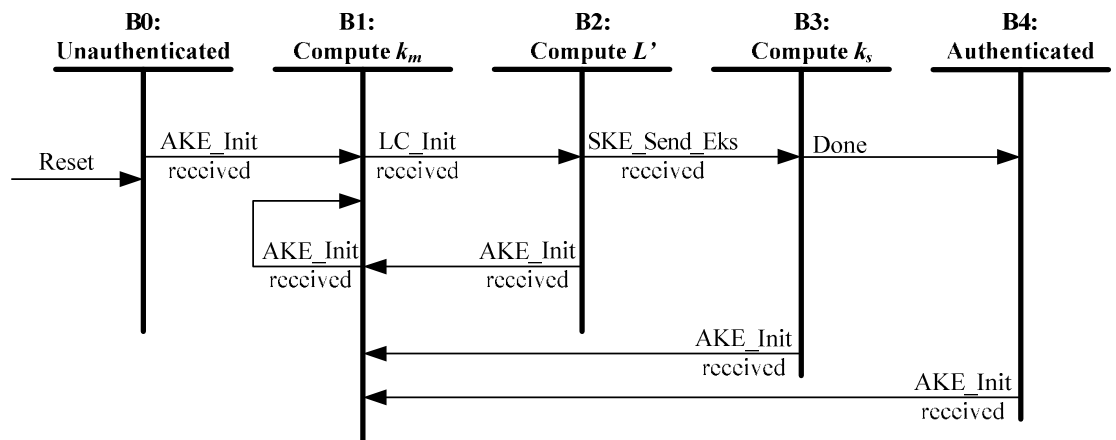


Figure 2.10. HDCP Receiver Authentication Protocol State Diagram

**Transition Any State:B0.** Reset conditions at the HDCP Receiver cause the HDCP Receiver to enter the unauthenticated state.

**State B0: Unauthenticated.** The HDCP Receiver is awaiting the reception of  $r_{tx}$  from the HDCP Transmitter to trigger the authentication protocol.

**Transition B0:B1.**  $r_{tx}$  is received as part of the AKE\_Init message from the HDCP Transmitter.

**State B1: Compute  $k_m$ .** In this state, the HDCP Receiver sends AKE\_Send\_Cert message in response to AKE\_Init, sends AKE\_Receiver\_Info message to the transmitter if AKE\_Transmitter\_Info message is received from the transmitter, generates and sends  $r_{tx}$  as part of AKE\_Send\_rrx message. If AKE\_No\_Stored\_km is received, it decrypts  $k_m$  with  $k_{priv_{tx}}$ , calculates  $H'$ . It sends AKE\_Send\_H\_prime message immediately after computation of  $H'$  to ensure that the message is received by the transmitter within the specified one second timeout at the transmitter.

If the HDCP Receiver does not receive AKE\_Transmitter\_Info message before the reception of AKE\_No\_Stored\_km or AKE\_Stored\_km message, it indicates that the HDCP Transmitter is an HDCP 2.0-compliant device.

If AKE\_Stored\_km is received, the HDCP Receiver decrypts  $E_{kh}(k_m)$  to derive  $k_m$  and calculates  $H'$ . It sends AKE\_Send\_H\_prime message immediately after computation of  $H'$  to ensure that the message is received by the transmitter within the specified 200 ms timeout at the transmitter.

If `AKE_No_Stored_km` is received, this is an indication to the HDCP Receiver that the HDCP Transmitter does not contain a  $k_m$  stored corresponding to its *Receiver ID*. It implements pairing with the HDCP Transmitter as explained in Section 2.2.1.

**Transition B1: B1.** Should the HDCP Transmitter send an `AKE_Init` while the HDCP Receiver is in State B1, the HDCP Receiver abandons intermediate results and restarts computation of  $k_m$ .

**Transition B1: B2.** The transition occurs when  $r_n$  is received as part of `LC_Init` message from the transmitter.

**State B2: Compute  $L'$ .** The HDCP Receiver computes  $L'$  required during locality check and sends `LC_Send_L_prime` message.

**Transition B2: B1.** Should the HDCP Transmitter send an `AKE_Init` while the HDCP Receiver is in State B2, the HDCP Receiver abandons intermediate results and restarts computation of  $k_m$ .

**Transition B2: B3.** The transition occurs when `SKE_Send_Eks` message is received from the transmitter.

**State B3: Compute  $k_s$ .** The HDCP Receiver decrypts  $E_{dkey}(k_s)$  to derive  $k_s$ .

**Transition B3: B1.** Should the HDCP Transmitter send an `AKE_Init` while the HDCP Receiver is in State B3, the HDCP Receiver abandons intermediate results and restarts computation of  $k_m$ .

**Transition B3: B4.** Successful computation of  $k_s$  transitions the receiver into the authenticated state.

**State B4: Authenticated.** The HDCP Receiver has completed the authentication protocol. Periodically, it updates its *coarseInputCtr* with the corresponding counter values received from the transmitter.

**Transition B4: B1.** Should the HDCP Transmitter send an `AKE_Init` while the HDCP Receiver is in State B4, the HDCP Receiver abandons intermediate results and restarts computation of  $k_m$ .

## 2.10 HDCP Repeater State Diagrams

The HDCP Repeater has one HDCP-protected Interface connection to an upstream HDCP Transmitter and one or more HDCP-protected Interface connections to downstream HDCP Receivers. The state diagram for each downstream connection (Figure 2. and Figure 2.) is substantially the same as that for the host HDCP Transmitter (Section 2.8), with two exceptions. First, the HDCP Repeater is not required to check for downstream Receiver IDs in a revocation list.

When the upstream HDCP-protected interface port of the HDCP Repeater is in an unauthenticated state, it signals the detection of an active downstream HDCP Receiver to the upstream HDCP Transmitter by propagating the `Rx AVReady` notification to the upstream HDCP Transmitter.

Whenever authentication is initiated by the upstream HDCP Transmitter by sending `AKE_Init`, the HDCP Repeater immediately initiates authentication on all its downstream HDCP-protected interface ports.

The HDCP Repeater may cache the latest Receiver ID list and topology information received from its downstream ports. Whenever authentication is attempted by the upstream transmitter by sending an  $r_{tx}$  value, the HDCP Repeater may propagate the cached Receiver ID list upstream without initiating a re-authentication on all its downstream ports. The HDCP Repeater must generate unique  $k_m$  values for HDCP Devices connected to each of its downstream HDCP-protected interface ports.

The HDCP Repeater may transmit the same Session Key,  $k_g$ , to all its authenticated and active downstream HDCP-protected Interface Ports, before beginning the transmission of HDCP Content to any of its downstream ports. After beginning the transmission of HDCP content by the HDCP Repeater to any of its downstream ports, subsequent connection of a new HDCP Receiver to its downstream port must result in (a) unique Session Key,  $k_g$ , exchanged with that HDCP Receiver or (b) a new authentication attempt with all its downstream HDCP-protected Interface Ports and subsequent exchange of the same Session Key,  $k_g$ , to all its authenticated and active downstream HDCP-protected Interface Ports.

If an HDCP Repeater has no active downstream HDCP devices, it must authenticate as an HDCP Receiver with REPEATER set to zero if it wishes to receive HDCP Content, but must not pass HDCP Content to downstream devices.

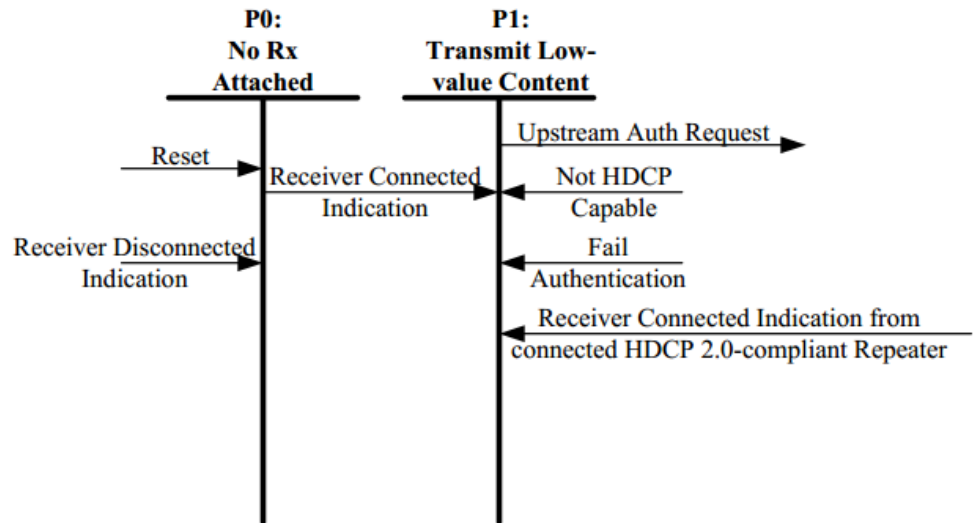
When the upstream HDCP-protected Interface Port of the HDCP Repeater transitions in to an unauthenticated state from an authenticated state (See Transition C5:C0 and Transition C6:C0 in Section 2.10.3), the HDCP Repeater must set the REAUTH\_REQ status bit in the *RxStatus* register. When the upstream HDCP Transmitter detects the REAUTH\_REQ status bit set by polling, it may initiate re-authentication with the HDCP Repeater with the transmission of a new AKE\_Init message.

### 2.10.1 Propagation of Topology Errors and AVReady / AVUnready Notification

**MAX\_DEVS\_EXCEEDED and MAX\_CASCADE\_EXCEEDED:** HDCP Repeaters must be capable of supporting DEVICE\_COUNT values less than or equal to 31 and DEPTH values less than or equal to 4. If the computed DEVICE\_COUNT for an HDCP Repeater exceeds 31, the error is referred to as MAX\_DEVS\_EXCEEDED error. The repeater sets MAX\_DEVS\_EXCEEDED = 'true' in the RepeaterAuth\_Send\_ReceiverID\_List message. If the computed DEPTH for an HDCP Repeater exceeds four, the error is referred to as MAX\_CASCADE\_EXCEEDED error. The repeater sets MAX\_CASCADE\_EXCEEDED = 'true' in the RepeaterAuth\_Send\_ReceiverID\_List message. When an HDCP Repeater receives a MAX\_DEVS\_EXCEEDED or a MAX\_CASCADE\_EXCEEDED error from a downstream HDCP Repeater, it must propagate the error to the upstream HDCP Transmitter and must not transmit  $V'$  and Receiver ID list.

### 2.10.2 HDCP Repeater Downstream State Diagram

In this state diagram and its following description, the downstream (HDCP Transmitter) side refers to the HDCP Transmitter functionality within the HDCP Repeater for its corresponding downstream HDCP-protected Interface Port.



Note: Transition arrows with no connected state (e.g. Reset) indicate transitions that can occur from multiple states

Figure 2.15. HDCP Repeater Downstream Link State Diagram

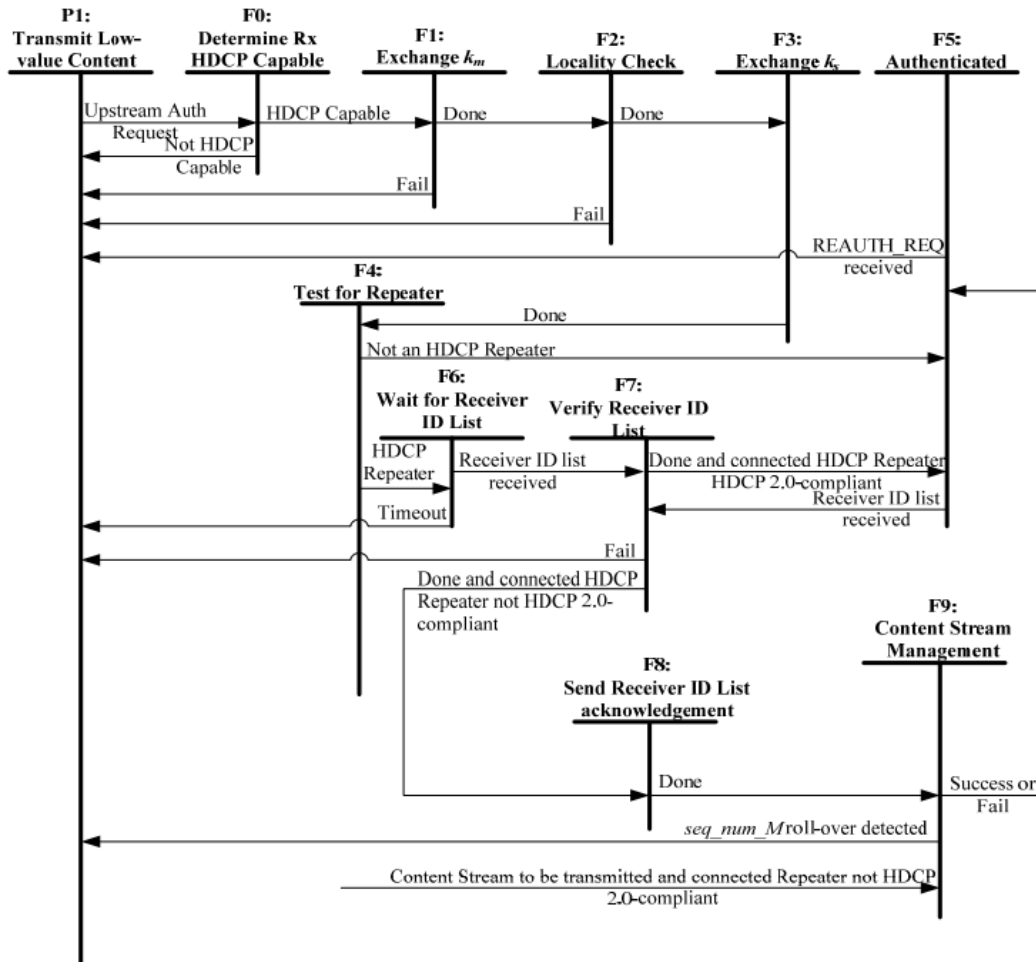


Figure 2.16. HDCP Repeater Downstream Authentication Protocol State Diagram

**Transition Any State:P0.** Reset conditions at the HDCP Repeater or disconnect (AVUnready notification) of all HDCP capable receivers cause the HDCP Repeater to enter the No Receiver Attached state. An AVUnready notification received from the connected downstream HDCP Repeater also causes this transition.

**Transition P0:P1.** The detection of a sink device (through Rx AVReady notification) indicates that the receiver is available and active (ready to display received content). When the receiver is no longer active, the downstream (HDCP Transmitter) side is notified through AVUnready notification.

**State P1: Transmit low-value content.** In this state the downstream side should begin sending the unencrypted video signal received from the upstream HDCP Transmitter with HDCP Encryption disabled. At any time a Rx AVReady notification received from the connected HDCP 2.0-compliant Repeater causes the downstream side to transition in to this state.

**Transition P1:F0.** Upon an Upstream Authentication Request, the downstream side should immediately attempt to determine whether the receiver is HDCP capable.



**State F0: Determine Rx HDCP Capable.** The downstream side determines that the receiver is HDCP capable as part of the setup and discovery procedures defined in the AVCL chapter of the WHDI specification. Since state F0 is reached upon an Upstream Authentication Request, authentication must be started immediately by the downstream side if the receiver is HDCP capable. A valid video screen is displayed to the user with encryption disabled during this time.

Note: The downstream side may initiate authentication before an Upstream Authentication Request is received.

**Transition F0:P1.** If the receiver is not HDCP capable, the downstream side continues to transmit low value content or informative on-screen display if low value content is received from the upstream HDCP Transmitter.

**Transition F0:F1.** If the receiver is HDCP capable, the downstream side initiates the authentication protocol.

**State F1: Exchange  $k_m$ .**

In this state, the downstream side initiates authentication by sending AKE\_Init message containing  $r_{tx}$  to the HDCP Receiver and sends AKE\_Transmitter\_Info message to the HDCP Receiver. It receives AKE\_Send\_Cert from the receiver containing REPEATER and  $cert_{rx}$ , and AKE\_Receiver\_Info message (if the HDCP Receiver is not HDCP 2.0-compliant). If the downstream side does not receive AKE\_Receiver\_Info message within 100 ms of the transmission of AKE\_Transmitter\_Info message, it indicates that the HDCP Receiver is an HDCP 2.0-compliant Device.

If the downstream side does not have  $k_m$  stored corresponding to the Receiver ID, it generates  $E_{k_{pub}}(k_m)$  and sends  $E_{k_{pub}}(k_m)$  as part of the AKE\_No\_Stored\_km message to the receiver after verification of signature on  $cert_{rx}$ . It receives AKE\_Send\_rrx message containing  $r_{rx}$  from the receiver. It computes H, receives AKE\_Send\_H\_prime message from the receiver containing  $H'$  within one second after sending AKE\_No\_Stored\_km to the receiver and compares  $H'$  against H.

If the downstream side has  $k_m$  stored corresponding to the Receiver ID, it sends AKE\_Stored\_km message containing  $E_{kh}(k_m)$  and  $m$  to the receiver and receives  $r_{rx}$  as part of AKE\_Send\_rrx message from the receiver. It computes H, receives AKE\_Send\_H\_prime message from the receiver containing  $H'$  within 200 ms after sending AKE\_Stored\_km to the receiver and compares  $H'$  against H.

If the downstream side does not have a  $k_m$  stored corresponding to the Receiver ID, it implements pairing with the HDCP Receiver as explained in Section 2.2.1.

**Transition F1:P1.** This transition occurs on failure of signature verification on  $cert_{rx}$  or if there is a mismatch between H and  $H'$ . This transition also occurs if AKE\_Send\_H\_prime message is not received within one second after sending AKE\_No\_Stored\_km or within 200 ms after sending AKE\_Stored\_km to the receiver.

**Transition F1:F2.** The downstream side implements locality check after successful completion of AKE and pairing.

**State F2: Locality Check.** In this state, the downstream side initiates locality check by sending LC\_Init message containing  $r_n$  to the HDCP Receiver, sets its watchdog timer to 7 ms and computes L. On receiving LC\_Send\_L\_prime message from the receiver, it compares  $L'$  against L.

**Transition F2:P1.**

This transition occurs on one or more consecutive locality check failures. Locality check fails when  $L'$  (or the most significant 128-bits of  $L'$ ) is not received within 7 ms and the watchdog timer at the downstream side expires or on a mismatch between  $L$  and  $L'$  (or the most significant 128-bits of  $L'$ ).

**Transition F2:F3.** The downstream side implements SKE after successful completion of locality check.

**State F3: Exchange  $k_s$ .** The downstream side sends encrypted Session Key,  $E_{dkey}(k_s)$ , and  $r_{iv}$  to the HDCP Receiver as part of the SKE\_Send\_Eks message. It enables HDCP Encryption 200 ms after sending encrypted Session Key. HDCP Encryption must be enabled only after successful completion of AKE, locality check and SKE stages.

**Transition F3:F4.** This transition occurs after completion of SKE.

**State F4: Test for Repeater.** The downstream side evaluates the REPEATER value that was received in State F1.

**Transition F4:F5.** REPEATER is 'false' (the HDCP Receiver is not an HDCP Repeater).

**State F5: Authenticated.** At this time, and at no prior time, the downstream side has completed the authentication protocol and is fully operational, able to deliver HDCP Content.

A periodic Link Synchronization is performed to maintain cipher synchronization between the downstream side and the HDCP Receiver.

**Transition F4:F6.** REPEATER is 'true' (the HDCP Receiver is an HDCP Repeater).

**State F6: Wait for Receiver ID List.** The downstream side sets up a three-second watchdog timer after sending SKE\_Send\_Eks.

**Transition F6:P1.** The watchdog timer expires before the RepeaterAuth\_Send\_ReceiverID\_List is received.

**Transition F6:F7.** RepeaterAuth\_Send\_ReceiverID\_List message is received.

**State F7: Verify Receiver ID List.**

If a transition in to this state occurs from State F6, the watchdog timer is cleared. If both MAX\_DEVS\_EXCEEDED and MAX\_CASCADE\_EXCEEDED are not 'true', computes  $V$ . If the connected HDCP Repeater is HDCP 2.0-compliant, compares  $V$  and  $V'$ . If the connected HDCP Repeater is not HDCP 2.0-compliant, compares the most significant 128-bits of  $V$  and  $V'$ . The Receiver IDs from this port are added to the Receiver ID list for this HDCP Repeater. The upstream HDCP Transmitter must be informed if topology maximums are exceeded.

**Transition F7:P1.**

This transition is made if a mismatch occurs between  $V$  and  $V'$  (if the connected HDCP Repeater is HDCP 2.0-compliant) or the most significant 128-bits of  $V$  and  $V'$  (if the connected HDCP Repeater is not HDCP 2.0-compliant). This transition is also made if the downstream side detects a roll-over of seq\_num\_V (if the repeater is not HDCP 2.0-compliant). A MAX\_CASCADE\_EXCEEDED or MAX\_DEVS\_EXCEEDED error also causes this transition.

**Transition F7:F5.** This transition is made if the connected HDCP Repeater is 2.0-compliant, on successful verification of  $V$  and  $V'$  and the downstream topology does not exceed specified maximums.

**Transition F7:F8.** This transition occurs if the connected HDCP Repeater is not HDCP 2.0 compliant, on successful verification of the most significant 128-bits of  $V$  and  $V'$ , the downstream side does not detect a roll-over of  $seq\_num\_V$  and the downstream topology does not exceed specified maximums.

**State F8:** Send Receiver ID list acknowledgment. The downstream code sends the least significant 128-bits of  $V$  to the HDCP receiver as part of the RepeaterAuth\_Send\_Ack message.

The RepeaterAuth\_Send\_Ack message must be received by the HDCP Repeater within one second from the transmission of the RepeaterAuth\_Send\_Ack message has been sent to the downstreamside.

**Transition F8:F9.** This transition occurs after the RepeaterAuth\_send\_ack message has been sent to the repeater.

**Transition F5:P1.** This transition occurs if a Receiver\_AuthStatus message with the REAUTH\_REQ set to "true" is received.

**Transition F5:F7.** This transition occurs whenever a RepeaterAuth\_Send\_ReceiverID\_List message is received from the connected HDCP repeater that is not HDCP 2.0-compliant.

**State F9:** Content Stream Management.

This stage is implemented if Content Stream is to be transmitted and the connected HDCP Repeater is not HDCP 2.0-compliant. The downstream side propagates the Content Stream management information, received from the upstream transmitter, using the RepeaterAuth\_Stream\_Manage message to the attached HDCP Repeater at least 100ms before the transmission of the corresponding Content Streams after HDCP Encryption. If the upstream transmitter is HDCP 2.0-compliant or HDCP 1.x-compliant, the downstream side will not receive the RepeaterAuth\_Stream\_Manage message from the upstream transmitter and assigns a Type value of 0x00 to all Content Streams received from the upstream transmitter and propagates the Content Stream management information using the RepeaterAuth\_Stream\_Manage message.

The downstream side must receive the RepeaterAuth\_Stream\_Ready message from the HDCP Repeater within 100 ms after the transmission of RepeaterAuth\_Stream\_Manage message and verifies  $M'$ . This step fails if the RepeaterAuth\_Stream\_Ready message is not received within 100 ms or if  $M$  is not equal to  $M'$ .

This stage may be implemented in parallel with the upstream propagation of topology information (State F4, State F6, State F7 and State F8) and with the flow of encrypted content and Link Synchronization (State F5). This state may be implemented asynchronously from the rest of the state diagram. A transition in to this state may occur from State F4, State F5, State F6, State F7 or State F8 if Content Stream is to be transmitted and the connected HDCP Repeater is not HDCP 2.0-compliant and the Content Stream management information is received from the upstream HDCP Transmitter. Also, the transition from State F9 must return to the appropriate state to allow for uninterrupted operation.

Note: Type 1 Content Streams must not be transmitted by the downstream side through its HDCP-protected Interface Ports connected to HDCP 1.x-compliant Devices and HDCP 2.0-compliant Repeaters.

**Transition F9:F5.** This transition occurs on success or failure of the Content Stream management stage.

**Transition F9: P1:** This transition occurs if  $seq\_num\_M$  rolls over.

Note: Since authentication with repeaters is implemented in parallel with the flow of encrypted content and Link Synchronization, the link synchronization process (i.e. State F5) must be implemented asynchronously from the rest of the state diagram. The transition into State F5 must occur from any state for which encryption is currently enabled. Also, the transition from state F5 returns to the appropriate state to allow for uninterrupted operation.

### 2.10.3 HDCP Repeater Upstream State Diagram

The HDCP Repeater upstream state diagram, illustrated in

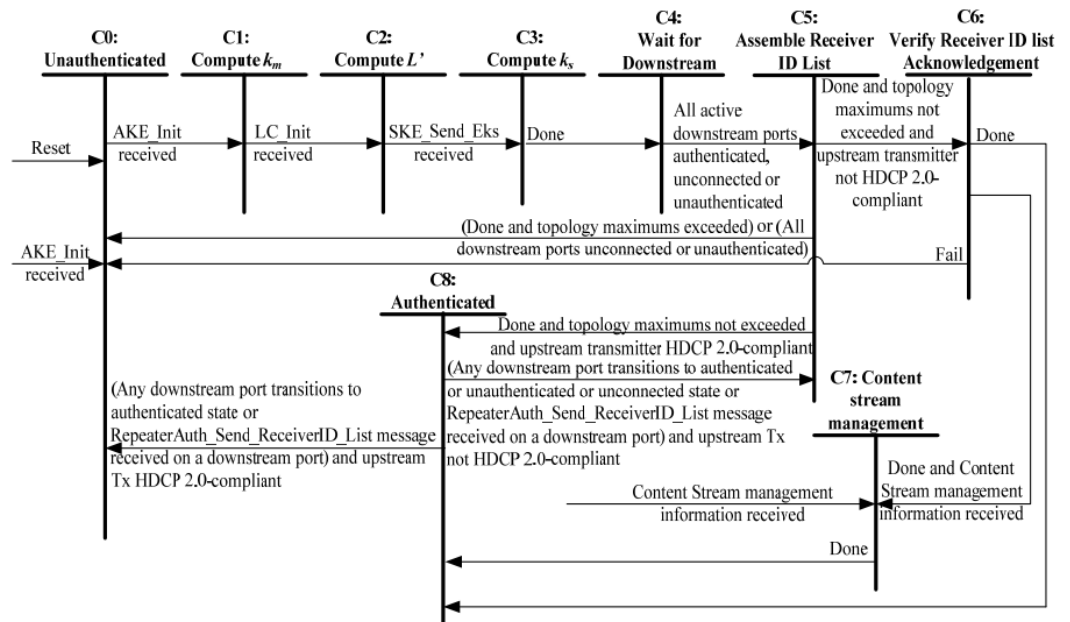
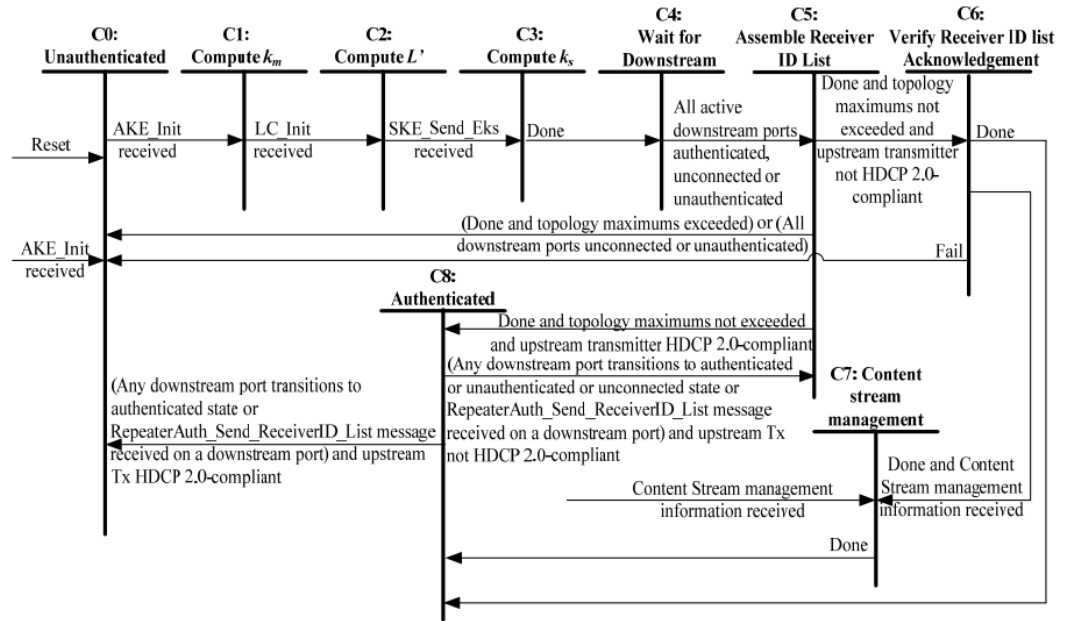


Figure 2., makes reference to states of the HDCP Repeater downstream state diagram. In this state diagram and its following description, the upstream (HDCP Receiver) side refers to the HDCP Receiver functionality within the HDCP Repeater for its corresponding upstream HDCP-protected Interface Port.



**Figure 2.17. HDCP Repeater Upstream Authentication Protocol State Diagram**

**Transitions Any State:C0.** Reset conditions at the HDCP Repeater cause the HDCP Repeater to enter the unauthenticated state. Re-authentication is forced any time AKE\_Init is received from the connected HDCP Transmitter, with a transition through the unauthenticated state.

**State C0: Unauthenticated.** The device is idle, awaiting the reception of  $r_{rx}$  from the HDCP Transmitter to trigger the authentication protocol.

**Transition C0:C1.**  $r_{rx}$  is received as part of the AKE\_Init message from the HDCP Transmitter.

**State C1: Compute  $k_m$ .**

In this state, the upstream (HDCP Receiver) side sends AKE\_Send\_Cert message in response to AKE\_Init, sends AKE\_Receiver\_Info message to the transmitter if AKE\_Transmitter\_Info message is received from the transmitter, generates and sends  $r_{rx}$  as part of AKE\_Send\_ $r_{rx}$  message. If AKE\_No\_Stored\_ $k_m$  is received, it decrypts  $k_m$  with  $k_{priv_{rx}}$ , calculates  $H'$ . It sends AKE\_Send\_H\_prime immediately after computation of  $H'$  to ensure that the message is received by the transmitter within the specified one second timeout at the transmitter.

If the upstream side does not receive AKE\_Transmitter Info message before the reception of AKE\_No\_Stored  $k_m$  or AKE\_Stored  $k_m$  message, it indicates that the HDCP Transmitter is an HDCP 2.0-compliant Device.

If AKE\_Stored\_ $k_m$  is received, the upstream side decrypts  $E_{k_{th}}(k_m)$  to derive  $k_m$  and calculates  $H'$ . It sends AKE\_Send\_H\_prime message immediately after computation of  $H'$  to ensure that the message is received by the transmitter within the specified 200 ms timeout at the transmitter.

If AKE\_No\_Stored\_ $k_m$  is received, this is an indication to the upstream side that the HDCP Transmitter does not contain a  $k_m$  stored corresponding to its Receiver ID. It implements pairing with the HDCP Transmitter as explained in Section 2.2.1.

**Transition C1:C2.** The transition occurs when  $r_n$  is received as part of LC\_Init message from the transmitter.

**State C2: Compute  $L'$ .** The upstream side computes  $L'$  required during locality check and sends LC\_Send\_L\_prime message.

**Transition C2: C3.** The transition occurs when SKE\_Send\_Eks message is received from the transmitter.

**State C3: Compute  $k_s$ .** The upstream side decrypts  $E_{dk_{key}}(k_s)$  to derive  $k_s$ .

**Transition C3: C4.** Successful computation of  $k_s$  causes this transition.

**State C4: Wait for Downstream.** The upstream state machine waits for all downstream HDCP-protected Interface Ports of the HDCP Repeater to enter the unconnected (State P0), inactive or unauthenticated (State P1), or the authenticated state (State F5).

**Transition C4:C5.** All downstream HDCP-protected Interface Ports with connected HDCP Receivers have reached the state of authenticated, unconnected or inactive state.

**State C5: Assemble Receiver ID List.**

The upstream side assembles the list of all connected downstream topology HDCP Devices as the downstream HDCP-protected Interface Ports reach terminal states of the authentication protocol. An HDCP-protected Interface Port that advances to State P0, the unconnected state, or P1, the unauthenticated state, does not add to the list. A downstream HDCP-protected Interface Port that arrives in State F5 that has an HDCP Receiver that is not an HDCP Repeater connected, adds the Receiver ID of the connected HDCP Receiver to the list. Downstream HDCP-protected Interface Ports that arrive in State F5 that have an HDCP Repeater connected will cause the *Receiver ID* list read from the connected HDCP Repeater, plus the *Receiver ID* of the connected HDCP Repeater itself, to be added to the list.

Note: The upstream side may add the *Receiver ID* list read from the HDCP Repeater connected to the downstream HDCP-protected Interface port, plus the *Receiver ID* of the connected HDCP Repeater itself to the list after the downstream port has transitioned in to State F8.

When the Receiver ID list for all downstream HDCP Receivers has been assembled, the upstream side computes DEPTH, DEVICE\_COUNT and the upstream  $V'$  and sends RepeaterAuth\_Send\_ReceiverID\_List message to the upstream HDCP Transmitter. In the case of a MAX\_DEVS\_EXCEEDED or a MAX\_CASCADE\_EXCEEDED error, it does not transmit  $V'$  (or the most significant 128-bits of  $V'$ ), DEPTH, DEVICE\_COUNT, Receiver ID list and if applicable, HDCP2\_LEGACY\_DEVICE\_DOWNSTREAM and HDCP1\_DEVICE\_DOWNSTREAM. When an HDCP Repeater receives a MAX\_DEVS\_EXCEEDED or MAX\_CASCADE\_EXCEEDED error from a downstream HDCP Repeater, it is required to inform the upstream HDCP Transmitter.

If any downstream port connected to an HDCP Repeater receives HDCP2\_LEGACY\_DEVICE\_DOWNSTREAM = 'true' or HDCP1\_DEVICE\_DOWNSTREAM = 'true', the upstream side sets the corresponding values to 'true' in the RepeaterAuth\_Send\_ReceiverID\_List message to the upstream HDCP Transmitter.

**Transition C5:C0.** This transition occurs if RepeaterAuth\_Send\_ReceiverID\_List message has been sent to the upstream HDCP Transmitter and topology maximums are exceeded i.e. on a MAX\_DEVS\_EXCEEDED or MAX\_CASCADE\_EXCEEDED error. This transition also occurs

if all downstream HDCP-protected Interface Ports have reached the state of unconnected or unauthenticated.

**Transition C5:C6.** RepeaterAuth\_Send\_ReceiverID\_List message has been sent to the upstream HDCP Transmitter and topology maximums are not exceeded and upstream transmitter is not HDCP 2.0-compliant.

**Transition C5:C8.** RepeaterAuth\_Send\_ReceiverID\_List message has been sent to the upstream HDCP Transmitter and topology maximums are not exceeded and upstream transmitter is HDCP 2.0-compliant.

**State C6.** Verify Receiver ID list acknowledgement. In this state, the upstream side receives the RepeaterAuth\_Send\_Ack message from the upstream transmitter and compares the least significant 128-bits of  $V$  and  $V'$ . A match between the least significant 128-bits of  $V$  and  $V'$  indicates successful upstream transmission of topology information. The RepeaterAuth\_Send\_Ack message must be received by the upstream side within one second from the transmission of the RepeaterAuth\_Send\_ReceiverID\_List message to the upstream transmitter if the transmitter is not HDCP 2.0-compliant.

**Transition C6:C0.** This transition occurs if the RepeaterAuth\_Send\_Ack message is not received by the upstream side within one second or on a mismatch between the least significant 128-bits of  $V$  and  $V'$ . If this transition occurs, the upstream side must send the Receiver\_AuthStatus message with the REAUTH REQ set to 'true' to the upstream transmitter.

**Transition C6:C7.** This transition occurs if the RepeaterAuth\_Send\_Ack message is received by the upstream side within one second, on a successful match between the least significant 128-bits of  $V$  and  $V'$  and if Content Stream management information is received from the upstream transmitter.

**Transition C6:C8.** This transition occurs if the RepeaterAuth\_Send\_Ack message is received by the upstream side within one second and on a successful match between the least significant 128-bits of  $V$  and  $V'$ .

**State C7: Content Stream Management.** On receiving the RepeaterAuth\_Stream\_Manage message, the upstream side computes  $M'$  and sends it to the upstream Transmitter as part of the RepeaterAuth\_Stream\_Ready message.

This stage may be implemented in parallel with the upstream propagation of topology information (State C4, State C5 and State C6) and with the flow of encrypted content and link synchronization (State C8). This state may be implemented asynchronously from the rest of the state diagram. A transition in to this state may occur from State C4, State C5, State C6 or State C8 if Content Stream management information is received from the upstream transmitter. Also, the transition from State C7 may return to the appropriate state to allow for uninterrupted operation.

The upstream side must be prepared to implement this stage in parallel with the upstream propagation of topology information and with the flow of encrypted content and link synchronization if these stages are implemented in parallel by the upstream transmitter.

**Transition C7:C8.** This transition occurs after RepeaterAuth\_Stream\_Ready message has been sent to the upstream transmitter.

**State C8: Authenticated.** The upstream side has completed the authentication protocol. Periodically, it updates its *inputCtr* with the value received from the transmitter.

**Transition C8:C5.** This transition occurs only if the upstream HDCP Transmitter is not HDCP 2.0-compliant and on detection of any changes to the topology.

This transition occurs when a downstream port that was previously in the unauthenticated (State P1) or unconnected (State P0) state transitions in to the authenticated (State F5) state. For example, the transition may occur when a new HDCP Receiver is connected to a downstream port, that previously had no receivers connected, and the downstream port completes the authentication protocol with the HDCP Receiver.

This transition also occurs when a downstream port that was previously in an authenticated state transitions in to an unauthenticated or unconnected state. For example, the transition may occur when an active, authenticated HDCP Receiver attached to the downstream port is disconnected.

Reception of a RepeaterAuth\_Send\_ReceiverID List message on a downstream port from the connected downstream HDCP Repeater also causes this transition.

**Transition C8:C0.** This transition occurs only if the upstream HDCP Transmitter is HDCP 2.0-compliant and on detection of any changes to the topology.

This transition occurs when a downstream port that was previously in the unauthenticated (State P1) or unconnected (State P0) state transitions in to the authenticated (State F5) state. For example, the transition may occur when a new HDCP Receiver is connected to a downstream port, that previously had no receivers connected, and the downstream port completes the authentication protocol with the HDCP Receiver.

Reception of a RepeaterAuth\_Send\_ReceiverID List message on a downstream port from the connected downstream HDCP Repeater also causes this transition.

If this transition occurs, the upstream side must propagate a Receiver Connected Indication to the upstream HDCP Transmitter.

Note: Since Link Synchronization may be implemented in parallel with the upstream propagation of topology information (State C4, State C5 and State C6) and Content Stream management (State C7), the link synchronization process (i.e. State C8) may be implemented asynchronously from the rest of the state diagram. The transition into State C8 may occur from any state for which encryption is currently enabled. Also, the transition from state C8 may return to the appropriate state to allow for uninterrupted operation.

The upstream side must be prepared to implement the link synchronization process in parallel with the upstream propagation of topology information and Content Stream management if these stages are implemented in parallel by the upstream transmitter.



## 2.11 Converters

### 2.11.1 HD​CP 2 – HD​CP 1.x Converters

HD​CP 2 – HD​CP 1.x converters are HD​CP Repeaters with an HD​CP 2 compliant interface port on the upstream (HD​CP Receiver) side and one or more HD​CP 1.x compliant interface ports on the downstream (HD​CP Transmitter) side.

The HD​CP 1.x compliant downstream side implements the state diagram explained in the corresponding HD​CP 1.x specification (See Section 1.5).

The HD​CP 2 compliant upstream side implements the state diagram as explained in Section 2.10.3 with these modifications.

- **State C5: Assemble Receiver ID List.** The upstream side assembles the list of all connected downstream topology HD​CP Devices as the downstream HD​CP-protected Interface Ports reach terminal states of the authentication protocol. An HD​CP-protected Interface Port that advances to the unconnected state or the unauthenticated state does not add to the list. A downstream HD​CP-protected Interface Port that arrives in an authenticated state that has an HD​CP Receiver that is not an HD​CP Repeater connected, adds the *Bksv* of the connected HD​CP Receiver to the Receiver ID list. Downstream HD​CP-protected Interface Ports that arrive in an authenticated state that have an HD​CP Repeater connected will cause the KSV list read from the connected HD​CP Repeater, plus the *Bksv* of the connected HD​CP Repeater itself, to be added to the list. KSVs are used in place of *Receiver IDs* and are added to the Receiver ID list in big-endian order

When the Receiver ID list (comprising KSVs of connected downstream HD​CP 1.x Receivers, where the KSVs are added to the list in big-endian order) for all downstream HD​CP Receivers has been assembled, the upstream side computes DEPTH, DEVICE COUNT and the upstream *V'* and sends RepeaterAuth\_Send\_ReceiverID List message to the upstream HD​CP Transmitter. In the case of a MAX\_DEVS\_EXCEEDED or a MAX\_CASCADE\_EXCEEDED error, it does not transmit *V'* (or the most significant 128-bits of *V'*), DEPTH, DEVICE COUNT, Receiver ID list and if applicable, HD​CP2\_LEGACY\_DEVICE\_DOWNSTREAM and HD​CP1\_DEVICE\_DOWNSTREAM. When an HD​CP Repeater receives a MAX\_DEVS\_EXCEEDED or MAX\_CASCADE\_EXCEEDED error from a downstream HD​CP Repeater, it is required to inform the upstream HD​CP Transmitter.

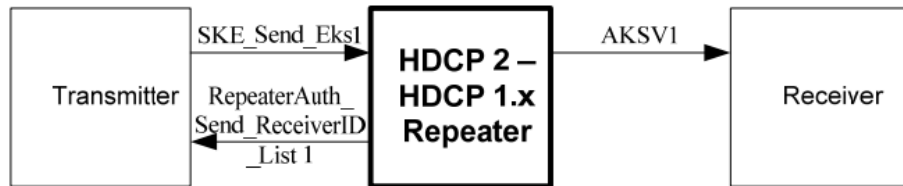


Figure 2.18. HD​CP2 – HD​CP 1.x Repeater Protocol Timing with Receiver Attached

From	To	Max Delay	Conditions and Comments
SKE_Send_Eks1 Session Key received from Upstream HDCP Transmitter	AKSV1 HDCP Repeater's <i>Aksv</i> transmitted downstream	100 ms	Downstream propagation time.
AKSV1 HDCP Repeater's <i>Aksv</i> transmitted downstream	RepeaterAuth_Send_ReceiverID_List1 <i>Receiver IDs</i> and topology information transmitted upstream	200 ms	Upstream propagation time when no downstream HDCP Repeaters are attached (no downstream KSV lists to process)

Table 2.3. HDCP2 – HDCP 1.x Repeater Protocol Timing with Receiver Attached

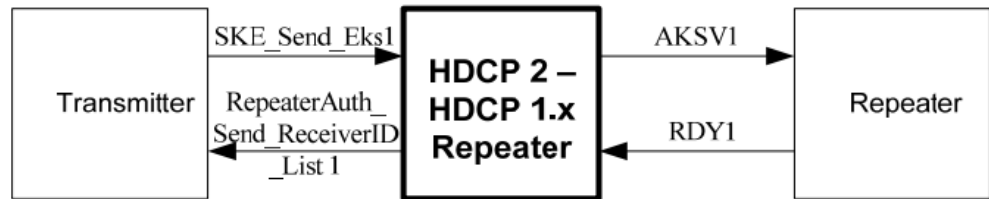


Figure 2.19. HDCP2 – HDCP 1.x Repeater Protocol Timing with Repeater Attached

From	To	Max Delay	Conditions and Comments
SKE_Send_Eks1 Session Key received from Upstream HDCP Transmitter	AKSV1 HDCP Repeater's <i>Aksv</i> transmitted downstream	100 ms	Downstream propagation time.
RDY1 Downstream <i>Receiver IDs</i> and topology information received	RepeaterAuth_Send_ReceiverID_List1 <i>Receiver IDs</i> and topology information transmitted upstream	200 ms	Upstream propagation time when one or more HDCP 1.x-compliant Repeaters are attached. From latest downstream READY. (downstream KSV lists must be processed)

Table 2.4. HDCP2 – HDCP 1.x Repeater Protocol Timing with Repeater Attached

### 2.11.2 HDCP 1.x – HDCP 2 Converters

HDCP 1.x — HDCP 2 converters are HDCP Repeaters with an HDCP 1.x compliant interface port on the upstream (HDCP Receiver) side and one or more HDCP 2 compliant interface ports on the downstream (HDCP Transmitter) side.

An HDCP1.x-HDCP2 Converter, that receives a DEVICE\_COUNT value equal to 1 and DEPTH equal to 3 from its downstream port, must set DEVICE\_COUNT value equal to 1 and DEPTH equal to 7 on its upstream port for transmission to the HDCP1.x-compliant upstream transmitter.

The HDCP 1.x compliant upstream side implements the state diagram explained in the corresponding HDCP 1.x specification (See Section 1.5).

The HDCP 2 compliant downstream side implements the state diagram as explained in Section 2.10.2 with these modifications.

- **State F7: Verify Receiver ID List.** If a transition in to this state occurs from State F6, the watchdog timer is cleared. If both MAX\_DEVS\_EXCEEDED and MAX\_CASCADE\_EXCEEDED are not 'true', computes  $V$ . If the connected HDCP Repeater is HDCP 2.0-compliant, compares  $V$  and  $V'$ . If the connected HDCP Repeater is not MCP 2.0-compliant, compares the most significant 128-bits of  $V$  and  $V'$ . The Receiver IDs from this port are used in place of KSVs and are added to the KSV list for this HDCP Repeater. KSV list is constructed by appending Receiver IDs in little-endian order. The upstream HDCP Transmitter must be informed if topology maximums are exceeded.

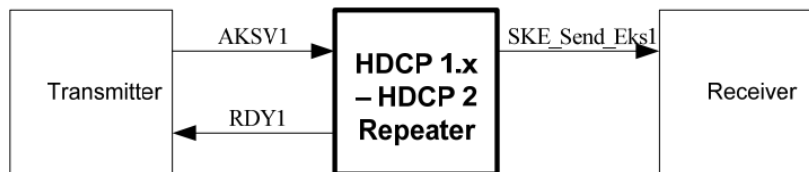
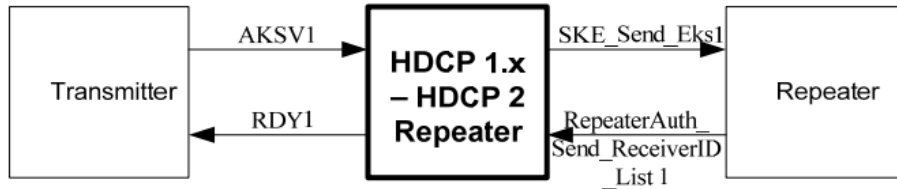


Figure 2.20. HDCP 1.x - HDCP2 Repeater Protocol Timing with Receiver Attached

From	To	Max Delay	Conditions and Comments
AKSV1 Upstream HDCP Transmitter $A_{ksv}$ received	SKE_Send_Eks1 $k_s$ generated by HDCP Repeater transmitted downstream	400 ms	Downstream propagation time.
SKE_Send_Eks1 $k_s$ generated by HDCP Repeater transmitted downstream	RDY1 Upstream READY asserted	500 ms	Upstream propagation time when no downstream HDCP Repeaters are attached (no downstream Receiver ID lists to process)

**Table 2.5. HDCP 1.x - HDCP2 Repeater Protocol Timing with Receiver Attached**



**Figure 2.21. HDCP 1.x - HDCP2 Repeater Protocol Timing with Repeater Attached**

From	To	Max Delay	Conditions and Comments
AKSV1 Upstream HDCP Transmitter <i>Aksv</i> received	SKE_Send_Eks1 <i>k<sub>s</sub></i> generated by HDCP Repeater transmitted downstream	400 ms	Downstream propagation time.
RepeaterAuth_Send_ReceiverID_List 1 Downstream <i>Receiver IDs</i> and topology information received	RDY1 Upstream READY asserted	500 ms	Upstream propagation time when one or more HDCP Repeaters are attached. From latest downstream RepeaterAuth_Send_ReceiverID_List message. (downstream Receiver ID lists must be processed)

**Table 2.6. HDCP 1.x - HDCP2 Repeater Protocol Timing with Repeater Attached**

## 2.12 Session Key Validity

When HDCP Encryption is disabled, the transmitter and receiver ceases to perform HDCP Encryption (Section 3.3) and stops incrementing the *coarseInputCtr*.

If HDCP Encryption was disabled, from its enabled state, due to the detection of AVReady notification, AVUnready notification or authentication failures, the HDCP Transmitter expires the Session Key. The HDCP Transmitter initiates re-authentication by the transmission of a new *r<sub>k</sub>*. In all other cases, where HDCP Encryption was disabled, from its enabled state, while the link was still active and authenticated (for e.g., HDCP Encryption may be briefly disabled during transmission of low value content), the HDCP Transmitter need not expire the Session Key. The HDCP Transmitter may maintain the encryption

parameters used during the HDCP Session i.e. *coarseInputCtr* value after the last HDCP Encryption operation (after which HDCP Encryption was disabled),  $k_s$  and  $r_{iv}$ . When re-enabled, HDCP Encryption is applied seamlessly, without requiring re-authentication, by using the same encryption parameters.

If HDCP Encryption was disabled, from its enabled state, the HDCP Receiver maintains  $k_s$  and  $r_{iv}$  used during the HDCP Session. If encryption was re-enabled, without intervening re-authentication requests from the transmitter, the HDCP Receiver uses the same  $k_s$  and  $r_{iv}$ . It updates its *coarseInputCtr* with the corresponding counter values received from the transmitter. (See Section 2.5.2 on Link Synchronization).

## 2.13 Random Number Generation

Random number generation is required both in the HDCP Transmitter logic and in the HDCP Receiver logic. Counter mode based deterministic random bit generator using AES-128 block cipher specified in NIST SP 800-90 is the recommended random number generator. The minimum entropy requirement for random values that are not used as secret key material (i.e.  $r_{tx}$ ,  $r_{rx}$ ,  $r_{iv}$ ,  $r_n$ ) is 40 random bits out of 64-bits. This means that a reasonable level of variability or entropy is established if out of 1,000,000 random ( $r_{tx}$ ,  $r_{rx}$ ,  $r_{iv}$  or  $r_n$ ) values collected after the first authentication attempt (i.e. after power-up cycles on the HDCP Transmitter or HDCP Receiver logic), the probability of there being any duplicates in this list of 1,000,000 random values is less than 50%.

For randomly generated secret key material ( $k_m$ ,  $k_s$ ) the minimum entropy requirement is 128-bits of entropy (i.e. the probability of there being any duplicates in the list of  $2^{64}$  secret values ( $k_m$  or  $k_s$ ) collected after power-up and first authentication attempt on the HDCP Transmitter logic is less than 50%).

A list of possible entropy sources that may be used for generation of random values used as secret key material include

- a true Random Number Generator or analog noise source, even if a poor (biased) one
- a pseudo-random number generator (PRNG), seeded by a true RNG with the required entropy, where the state is stored in non-volatile memory after each use. The state must be kept secret. Flash memory or even disk is usable for this purpose as long as it is secure from tampering.

A list of possible entropy sources that may be used for generation of random values not used as secret key material include

- timers, network statistics, error correction information, radio/cable television signals, disk seek times, etc.

a reliable (not manipulatable by the user) calendar and time-of-day clock. For example, some broadcast content sources may give reliable date and time information.

### 3 HDCP Encryption

#### 3.1 Description

Figure 3.1 shows how HDCP fits in to the WHDI protocol stack. The wireless link consists of two constituent links: a unidirectional high-speed stream transporting the AV content and a lower-speed bidirectional link used for control / status.

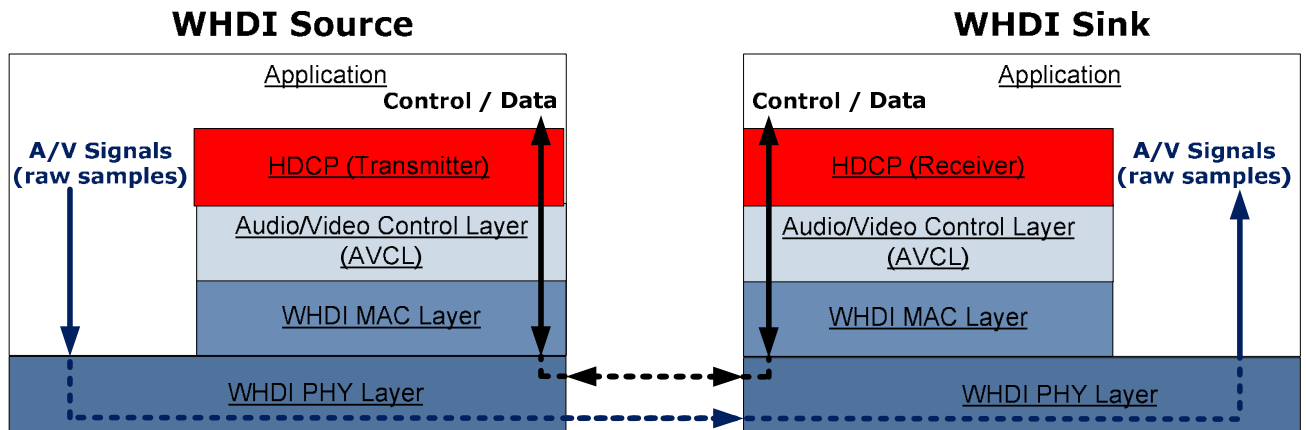


Figure 3.1. WHDI with HDCP Block Diagram (Informative)

Video in the HDCP Transmitter is assumed to be a stream of uncompressed pixel samples. Together with any associated audio or data streams, they are carried over the wireless channel as two streams of information:

- Coarse stream – represented by a stream of bytes
- Fine stream – represented by a stream of complex numbers

Device discovery, registration and association (including link setup and link release), is as described in the *WHDI 1.0 Specification*.

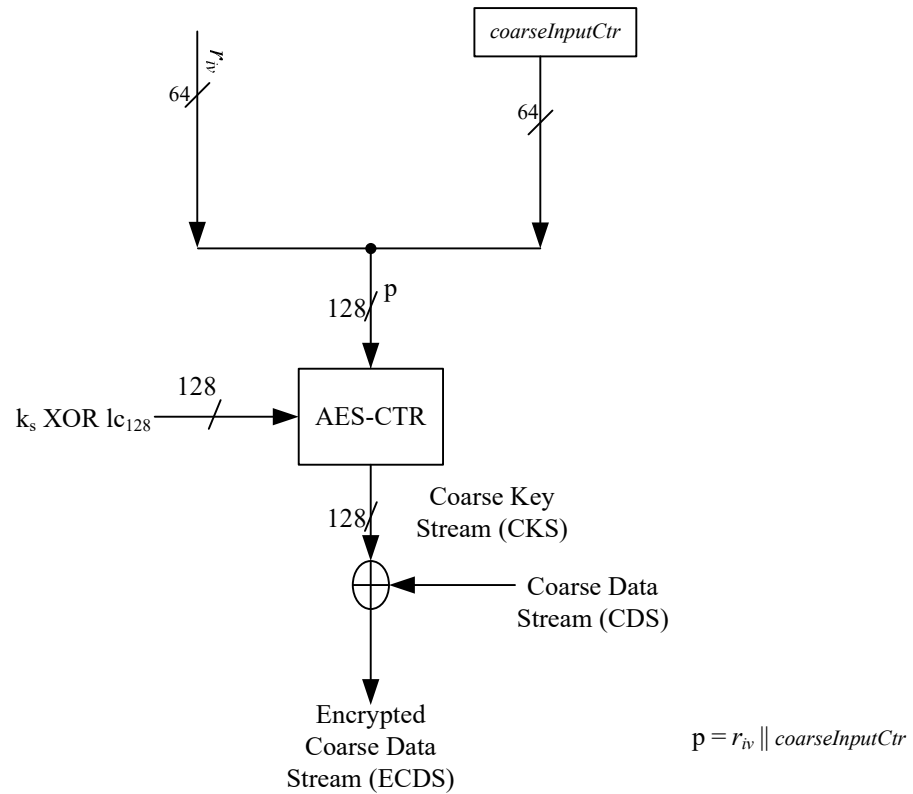
#### 3.2 AV Stream

WHDI AV streams consist of a coarse data stream and a fine data stream. The coarse data stream contains audio, video and control information. Both the coarse stream and the fine stream are subject to HDCP Encryption.

#### 3.3 HDCP Cipher

##### 3.3.1 HDCP Cipher for Coarse Stream

The HDCP Cipher for coarse data consists of a 128-bit AES module that is operated in a Counter (CTR) mode as illustrated in Figure 3.2.



**Figure 3.2. HDCP Cipher Structure for Coarse Data**

$k_s$  is the 128-bit Session Key which is XORed with  $lc_{128}$ .

$p = r_{iv} \parallel coarseInputCtr$ . All values are in big-endian order.

*coarseInputCtr* is a 64-bit counter. It is initialized to zero after SKE and must not be reset at any other time.

HDCP Encryption must be applied to WHDI Coarse payload data; WHDI Headers must not be encrypted.

During HDCP Encryption, the key stream produced by the AES-CTR module is XORed with 128-bit (16 Byte) block of coarse data to produce the 128-bit encrypted output. *coarseInputCtr* is incremented by one following encryption of every 128-bit block of coarse data for that frame. The value of *coarseInputCtr* must never be reused for a given set of encryption parameters i.e.  $k_s, r_{iv}$ .

The 16 Byte encryption block boundary must be aligned with the start of frame data. If the last block in an encrypted frame is less than 16 Bytes, only the encrypted payload bytes must be transmitted; i.e., the unused key stream bits produced by the AES-CTR module must be discarded, and not carried over to a subsequent frame.

Byte ordering is such that the least-significant byte of the 16 bytes (128-bit) key stream produced by AES-CTR module is XORed with the first byte in time in the 16 Byte payload data block.

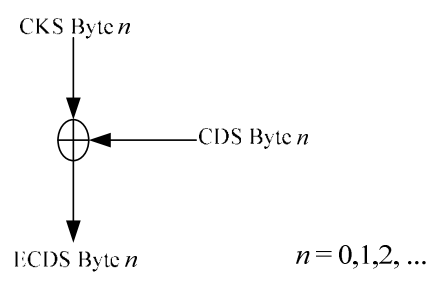
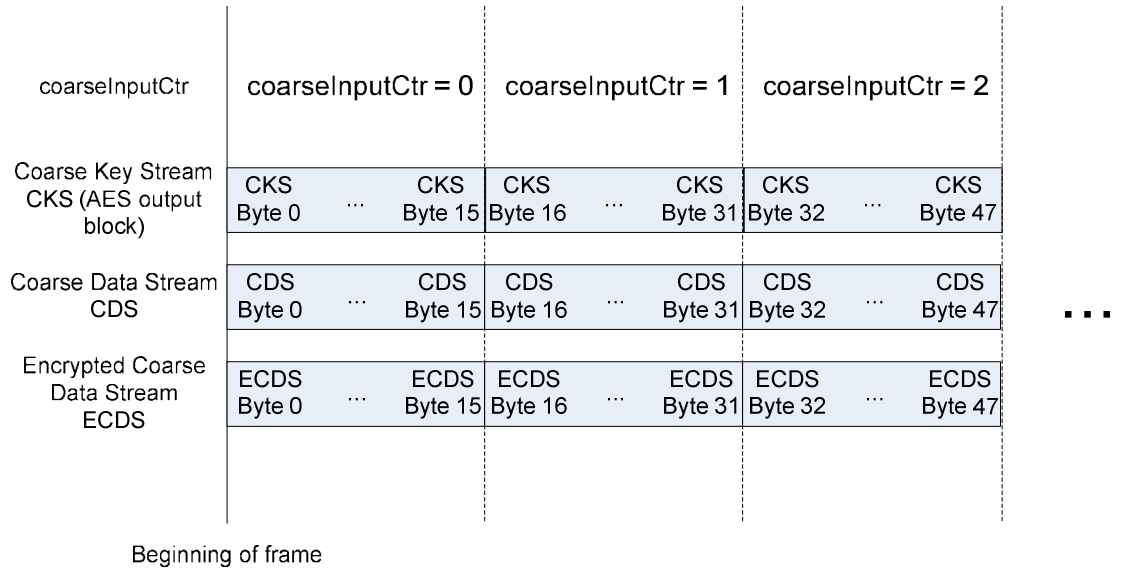


Figure 3.3. Byte Ordering in Coarse Data Encryption

### 3.4 HDCP Encryption Indication

Any WHDI frame containing an HDCP encrypted payload must include the 64 bit *coarseInputCtr* field in its Extended Header. The presence of these fields in the WHDI Extended Header structure serves to indicate that HDCP Encryption is enabled and the frame payload is encrypted. When HDCP Encryption is disabled, the *coarseInputCtr* field is not included in the WHDI Extended Header.

The *coarseInputCtr* is loaded at the beginning of each frame with the *coarseInputCtr* value transmitted in the Extended Header, thus maintaining link synchronization. As a result, the *coarseInputCtr* value transmitted in the frame header is used to generate the first 16 bytes coarse key stream for that frame, used to encrypt the first 16 bytes of the coarse data payload for that frame. HDCP Cipher Block



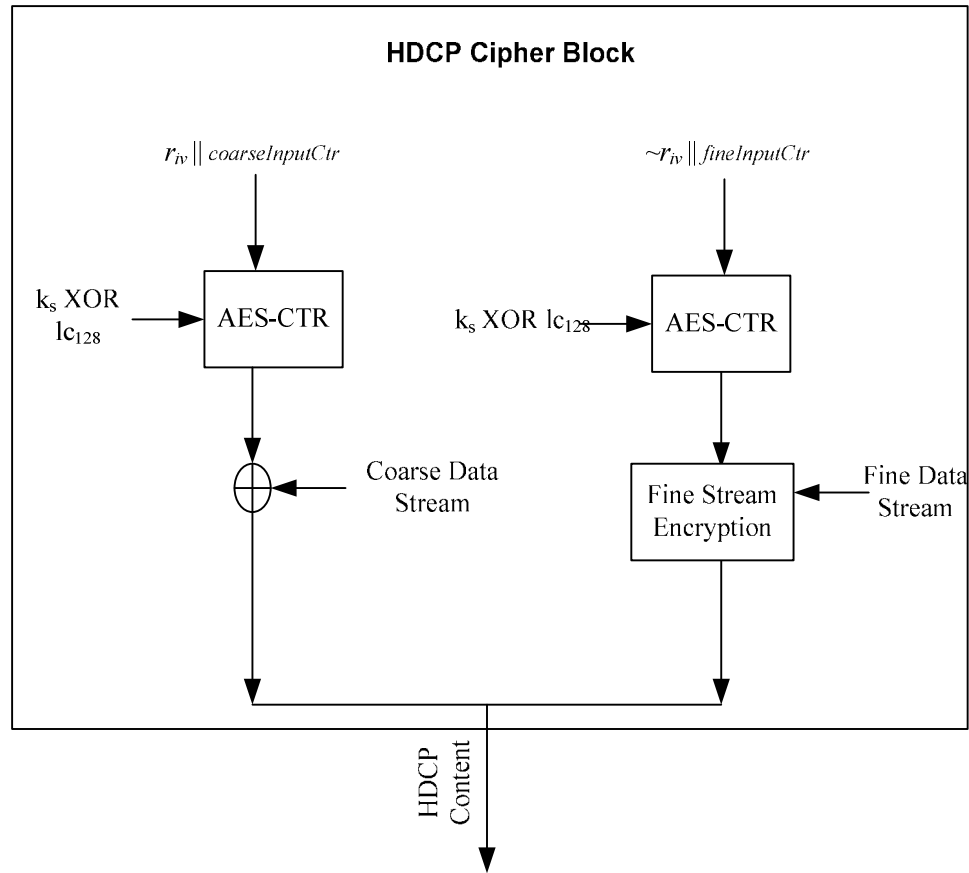


Figure 3.4. HDCP Cipher Block

### 3.5 Uniqueness of $k_s$ and $r_{iv}$

HDCP Receivers and HDCP Repeaters with multiple inputs may share the same Public Key Certificates and Private Keys across all inputs. The HDCP Transmitter (including downstream side of HDCP Repeater) must negotiate distinct  $k_m$  with each directly connected downstream HDCP Device. While  $r_{ix}$  used during each HDCP Session is required to be fresh, transmitters with multiple downstream HDCP links must ensure that each link receives a distinct  $r_{ix}$  value.

As illustrated in Figure 3.5, HDCP Transmitters, including downstream side of HDCP Repeaters, with multiple downstream HDCP links may share the same  $k_s$  and  $r_{iv}$  across those links only if HDCP Content from the same HDCP Cipher block is transmitted to those links.

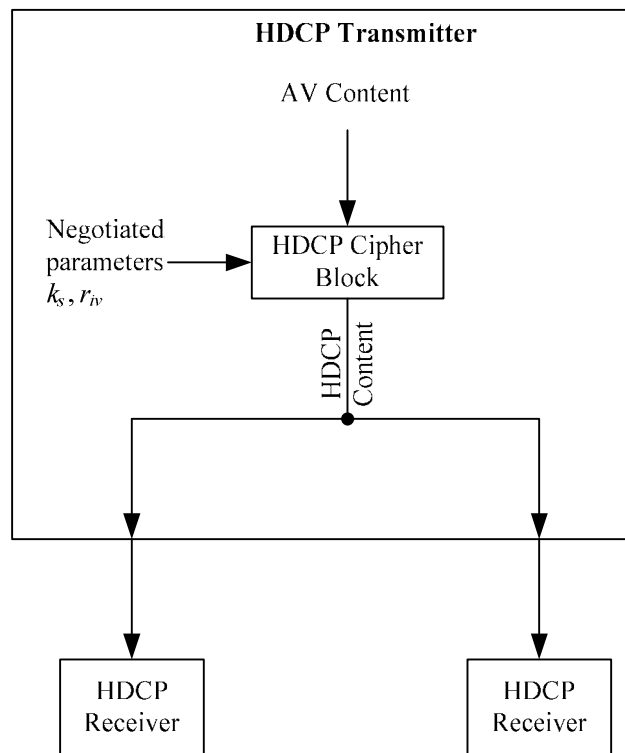


Figure 3.5.  $k_s$  and  $r_{iv}$  Shared across HDCP Links

As illustrated in Figure 3.6, HDCP Transmitters, including downstream side of HDCP Repeaters, with multiple downstream HDCP links must ensure that each link receives distinct  $k_s$  and  $r_{iv}$  values if HDCP Content from different HDCP Cipher blocks is transmitted to those links.

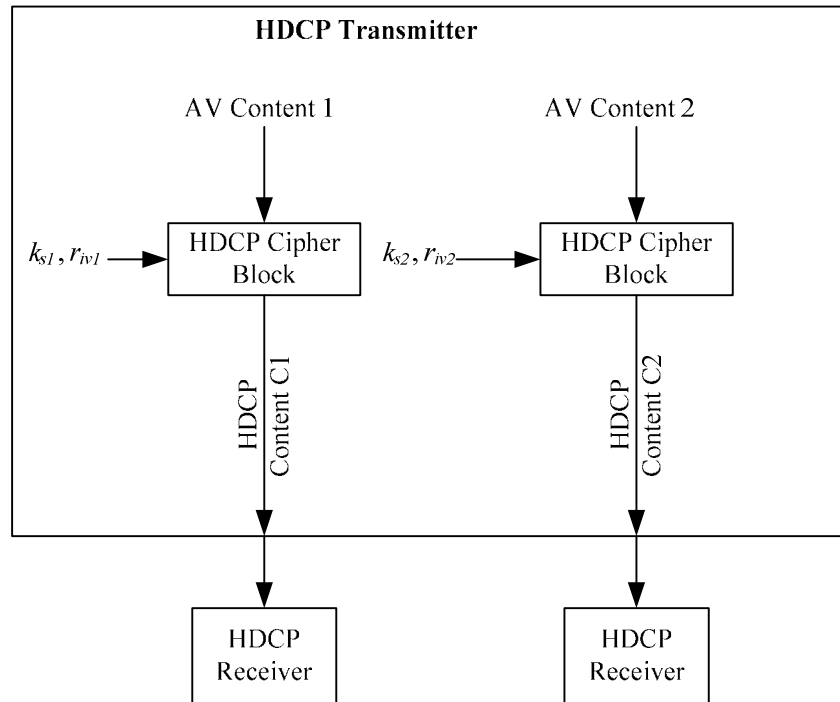


Figure 3.6. Unique  $k_s$  and  $r_{iv}$  across HDCP Links

## 4 Authentication Protocol Messages

### 4.1 HDCP Messages in WHDI

WHDI provides a bi-directional message delivery mechanism over the bi-directional control channel. Each message begins with a message header, containing a message type. (Please refer to chapter 7 of the WHDI specification for a complete description of message structure).

All HDCP messages shall be transmitted with a type equal to 0xDC in the message header.

The payload of an HDCP message shall be structured as described in Table 4.1.

Field	Size in Bytes	Description
Message ID	1	HDCP protocol message identifier, as described in Table 4.2
Parameters	Variable (message dependent)	Parameters specific to each message as described in Section 4.2

**Table 4.1. HDCP Message Payload Structure**

The payload of an HDCP message commences with a msg\_id specifying the message type, followed by parameters specific to each message. Parameter values spanning more than one byte must be sent most-significant byte first.

Valid values of msg\_id are shown in Table 4.2.

Message Type	msg_id Value
Null message	1
AKE Init	2
AKE Send Cert	3
AKE No Stored km	4
AKE Stored km	5
AKE Send rrx	6
AKE Send H_prime	7
AKE Send Pairing Info	8
LC Init	9
LC Send L_prime	10
SKE Send Eks	11
RepeaterAuth Send ReceiverID List	12
RTT Ready	13
RTT Challenge	14
RepeaterAuth Send Ack	15
RepeaterAuth Stream Manage	16
RepeaterAuth Stream Ready	17
Receiver AuthStatus	18
AKE Transmitter info	19
AKE Receiver info	20
Reserved	21-31

**Table 4.2. Values for msg\_id**

Note:

- The use of the Null message and Reserved values for msg\_id are not defined in this specification. HDCP Devices must be capable of receiving Null message and messages with reserved msg\_id values and must ignore these messages.

#### 4.2 Message Format

An identifier is associated with each field to indicate the numeric/logical value represented by the field:

- uint – unsigned integer in big endian format
- bool – a parameter one byte in length. The parameter is ‘true’ if the least-significant bit is non-zero, and false otherwise.

##### 4.2.1 AKE\_Init (Transmitter to Receiver)

Field	No. of Bytes	Identifier
msg_id	1	uint
$r_{rx}$	8	uint

Table 4.3. AKE\_Init Payload

##### 4.2.2 AKE\_Send\_Cert (Receiver to Transmitter)

The HDCP Receiver sets REPEATER to ‘true’ if it is an HDCP Repeater and ‘false’ if it is an HDCP Receiver that is not an HDCP Repeater. When REPEATER = ‘true’, the HDCP Receiver supports downstream connections as permitted by the Digital Content Protection LLC license.

Field	No. of Bytes	Identifier
msg_id	1	uint
REPEATER	1	bool
$cert_{rx}$	522	uint

Table 4.4. AKE\_Send\_Cert Payload

##### 4.2.3 AKE\_No\_Stored\_km (Transmitter to Receiver)

Field	No. of Bytes	Identifier
msg_id	1	uint
$E_{k_{pub}} k_m$	128	uint

Table 4.5. AKE\_No\_Stored\_km Payload

##### 4.2.4 AKE\_Stored\_km (Transmitter to Receiver)

Field	No. of Bytes	Identifier
msg_id	1	uint
$E_{kh} k_m$	16	uint
$m$	16	uint

Table 4.6. AKE\_Stored\_km Payload

##### 4.2.5 AKE\_Send\_rrx (Receiver to Transmitter)

Field	No. of Bytes	Identifier
msg_id	1	uint
$r_{rx}$	8	uint

Table 4.7. AKE\_Send\_rrx Payload

#### 4.2.6 AKE\_Send\_H\_prime (Receiver to Transmitter)

Field	No. of Bytes	Identifier
msg_id	1	uint
$H'$	32	uint

Table 4.8. AKE\_Send\_H\_prime Payload

#### 4.2.7 AKE\_Send\_Pairing\_Info (Receiver to Transmitter)

Field	No. of Bytes	Identifier
msg_id	1	uint
$E_{kh}, k_m$	16	uint

Table 4.9. AKE\_Send\_Pairing\_Info Payload

#### 4.2.8 LC\_Init (Transmitter to Receiver)

Field	No. of Bytes	Identifier
msg_id	1	uint
$r_n$	8	uint

Table 4.10. LC\_Init Payload

#### 4.2.9 LC\_Send\_L\_prime (Receiver to Transmitter)

If the upstream HDCP Transmitter is HDCP 2.0-compliant or the TRANSMITTER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit received as part of the AKE\_Transmitter Info message is set to zero or the receiver has set the RECEIVER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit to zero in its AKE Receiver Info message, the HDCP Receiver constructs the LC\_Send\_L\_prime message as given below.

Field	No. of Bytes	Identifier
LC_Send_L_prime {msg_id L' [255..0]}	1	uint
	32	uint

Table 4.11. LC\_Send\_L\_prime Payload

If the TRANSMITTER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit received as part of the AKE\_Transmitter Info message is set to one or the receiver has set the RECEIVER\_LOCALITY\_PRECOMPUTE\_SUPPORT bit to one in its AKE Receiver Info message, the HDCP Receiver constructs the LC\_Send\_L\_prime message as given below.

Field	No. of Bytes	Identifier
LC_Send_L_prime {msg_id L' [255..128]}	1	uint
	16	uint

Table 4.12. LC\_Send\_L\_prime Payload

#### 4.2.10 SKE\_Send\_Eks (Transmitter to Receiver)

The HDCP Transmitter constructs the SKE\_Send\_Eks message as given in Table 4.13 if the HDCP Receiver is compliant with HDCP 2.3 or higher, else the HDCP Transmitter constructs the SKE\_Send\_Eks message as given in Table 4.14

Field	No. of Bytes	Identifier
msg_id	1	uint
$E_{dkey} k_s [127..0]$	16	uint
$r_{iv} [63..0]$	8	uint
$HMAC(r_{iv}) [255..0]$	32	uint

Table 4.13. SKE\_Send\_Eks Payload

Field	No. of Bytes	Identifier
msg_id	1	uint
$E_{dkey} k_s [127..0]$	16	uint
$r_{iv} [63..0]$	8	uint

Table 4.14. SKE\_Send\_Eks Payload

#### 4.2.11 RepeaterAuth\_Send\_ReceiverID\_List (Receiver to Transmitter)

The HDCP Repeater constructs the RepeaterAuth\_Send\_ReceiverID\_List message as given in Table 4.14 if the upstream HDCP Transmitter is an HDCP 2.0-compliant Device, else the HDCP Repeater constructs the message as given in Table 4.15.

Receiver ID list is constructed by appending Receiver IDs in big-endian order.

Receiver ID list = Receiver ID0 || Receiver ID1 || ... Receiver IDn-1, where n is the DEVICE\_COUNT.

If the computed DEVICE\_COUNT for an HDCP Repeater exceeds 31, the repeater sets MAX\_DEVS\_EXCEEDED = 'true'. If the computed DEPTH for an HDCP Repeater exceeds four, the repeater sets MAX\_CASCADE\_EXCEEDED = 'true'. If topology maximums are not exceeded, MAX\_DEVS\_EXCEEDED = 'false' and MAX\_CASCADE\_EXCEEDED = 'false'.

The HDCP Repeater sets HDCP2\_LEGACY\_DEVICE\_DOWNSTREAM = 'true' if an HDCP 2.0-compliant repeater is attached to any one of its downstream ports, else it sets HDCP2\_0\_REPEATER\_DOWNSTREAM = 'false'.

The HDCP Repeater sets HDCP1\_DEVICE\_DOWNSTREAM = 'true' if an HDCP 1.x-compliant Device i.e. an HDCP 1.x-compliant Receiver or an HDCP 1.x-compliant Repeater is attached to any one of its downstream ports, else it sets HDCP1\_DEVICE\_DOWNSTREAM = 'false'.

Syntax	No. of Bytes	Identifier
RepeaterAuth_Send_ReceiverID_List{		
msg_id	1	uint
MAX_DEVS_EXCEEDED	1	bool
MAX_CASCADE_EXCEEDED	1	bool
if (MAX_DEVS_EXCEEDED != 1 && MAX_CASCADE_EXCEEDED != 1) {		
DEVICE_COUNT	1	uint
DEPTH	1	uint
V[255..0]	32	uint
for (j=0; j< DEVICE_COUNT; j++) {		
Receiver_IDj[39..0]	5	uint
}		
}		
}		

Table 4.15. RepeaterAuth\_Send\_ReceiverID\_List Payload

Syntax	No. of Bytes	Identifier
RepeaterAuth_Send_ReceiverID_List{		
msg_id	1	uint
MAX_DEVS_EXCEEDED	1	bool
MAX_CASCADE_EXCEEDED	1	bool
if (MAX_DEVS_EXCEEDED != 1 &&		
MAX_CASCADE_EXCEEDED != 1) {		
DEVICE_COUNT	1	uint
DEPTH	1	uint
HDCP2_0_REPEATER_DOWNSTREAM	1	bool
HDCP1_DEVICE_DOWNSTREAM	1	bool
seq_num_V	3	uint
V[255..128]	16	uint
for (j=0; j< DEVICE_COUNT; j++) {		
Receiver_IDj[39..0]	5	uint
}		
}		
}		

Table 4.16. RepeaterAuth\_Send\_ReceiverID\_List Payload

#### 4.2.12 RTT\_Ready (Receiver to Transmitter)

Field	No. of Bytes	Identifier
msg_id	1	uint

Table 4.17. RTT\_Ready Payload

#### 4.2.13 RTT\_Challenge (Transmitter to Receiver)

Field	No. of Bytes	Identifier
msg_id	1	uint
L[127:0]	16	uint

Table 4.18. RTT\_Challenge Payload

#### 4.2.14 Receiver\_AuthStatus (Receiver to Transmitter)

LENGTH parameter is the size of the Receiver\_AuthStatus message in bytes. An HDCP 2.2-compliant Receiver will set the LENGTH parameter equal to four bytes i.e. the combined size of the msg\_id, LENGTH and REAUTH\_REQ parameters. An HDCP 2.2-compliant transmitter that receives a Receiver\_AuthStatus message with the LENGTH parameter greater than four bytes must read the msg\_id, LENGTH and REAUTH\_REQ parameters and must ignore the remaining parameters.



Field	No. of Bytes	Identifier
msg_id	1	uint
LENGTH	2	uint
REAUTH_REQ	1	bool

**Table 4.19. Receiver\_AuthStatus Payload**

#### 4.2.15 AKE\_Transmitter\_Info (Transmitter to Receiver)

LENGTH parameter is the size of the AKE\_Transmitter\_Info message in bytes. An HDCP 2.2-compliant Transmitter will set the LENGTH parameter equal to six bytes i.e. the combined size of the msg\_id, LENGTH, VERSION and TRANSMITTER\_CAPABILITY\_MASK parameters. An HDCP 2.2-compliant Receiver that receives an AKE\_Transmitter\_Info message with the LENGTH parameter greater than six bytes must read the msg\_id, LENGTH, VERSION and TRANSMITTER\_CAPABILITY\_MASK parameters and must ignore the remaining parameters.

The HDCP Transmitter must set VERSION to 0x02.

Field	No. of Bytes	Identifier
msg_id	1	uint
LENGTH	2	uint
VERSION	1	uint
TRANSMITTER_CAPABILITY_MASK	2	uint

**Table 4.20. AKE\_Transmitter\_Info Payload**

Parameter	No. of Bytes	Description
TRANSMITTER_CAPABILITY_MASK	2	Bits 15:2: Reserved zeros. Bit 1: TRANSMITTER_CONTENT_CATEGORY_SUPPORT. This bit must be set to 1 by the HDCP Transmitter. Bit 0: TRANSMITTER_LOCALITY_PRECOMPUTE_SUPPORT. When this bit is set to one, it indicates that the HDCP Transmitter supports pre-computation of L during the locality check protocol.

**Table 4.21 TRANSMITTER\_CAPABILITY\_MASK Parameter**

#### 4.2.16 AKE\_Receiver\_Info (Receiver to Transmitter)

LENGTH parameter is the size of the AKE\_Receiver\_Info message in bytes. An HDCP 2.2-compliant Receiver will set the LENGTH parameter equal to six bytes i.e. the combined size of the msg\_id, LENGTH, VERSION and RECEIVER\_CAPABILITY\_MASK parameters. An HDCP 2.2-compliant transmitter that receives an AKE\_Receiver\_Info message with the LENGTH parameter greater than six bytes must read the msg\_id, LENGTH, VERSION and RECEIVER\_CAPABILITY\_MASK parameters and must ignore the remaining parameters.

The HDCP Receiver must set VERSION to 0x02.

Field	No. of Bytes	Identifier
msg_id	1	uint
LENGTH	2	uint
VERSION	1	uint
RECEIVER_CAPABILITY_MASK	2	uint

**Table 4.22. AKE\_Transmitter\_Info Payload**

Parameter	No. of Bytes	Description
RECEIVER_CAPABILITY_MASK	2	Bits 15:1: Reserved zeros. Bit 0: RECEIVER_LOCALITY_PRECOMPUTE_SUPPORT. When this bit is set to one, it indicates that the HDCP Receiver supports pre-computation of <i>L'</i> during the locality check protocol.

**Table 4.23. RECEIVER\_CAPABILITY\_MASK Parameter**

#### 4.2.17 RepeaterAuth\_Send\_Ack (Transmitter to Receiver)

Syntax	No. of Bytes	Identifier
RepeaterAuth_Send_Ack{ msg_id I[127..0] }	1 16	uint uint

**Table 4.24. RepeaterAuth\_Send\_Ack Payload**

#### 4.2.18 RepeaterAuth\_Stream\_Manage (Transmitter to Receiver)

Content Streams are assigned a Type value by the most upstream HDCP Transmitter based on instructions received from the Upstream Content Control Function.

STREAMID\_TYPE = ContentStreamID<sub>1</sub> || Type || ContentStreamID<sub>2</sub> || Type || ... || ContentStreamID<sub>k</sub> || Type

Parameter	No. of Bytes	Description
ContentStreamID	2	Content Stream Identification. This parameter must be assigned the PDI values as follows. ContentStreamID [15:3] = 0 ContentStreamID [2:0] = PID
Type	1	0x00 : Type 0 Content Streams. May be transmitted by the HDCP Repeater to all HDCP Devices. 0x01 : Type 1 Content Streams. Must not be transmitted by the HDCP Repeater to HDCP 1.x-compliant Devices, HDCP 2.0-compliant Devices and HDCP 2.1-compliant Devices. 0x02 - 0xFF : Reserved for future use only. Content Streams with reserved Type values must be treated similar to Type 1 Content Streams

**Table 4.25. RepeaterAuth\_Stream\_Manage Parameters**

#### 4.2.19 RepeaterAuth\_Stream\_Ready (Transmitter to Receiver)

Syntax	No. of Bytes	Identifier
<pre> RepeaterAuth_Stream_Ready{     msg_id     M'[255..0] }                     </pre>	<p>1</p> <p>32</p>	<p>uint</p> <p>uint</p>

**Table 4.26. RepeaterAuth\_Stream\_Ready Payload**

## 5 Renewability

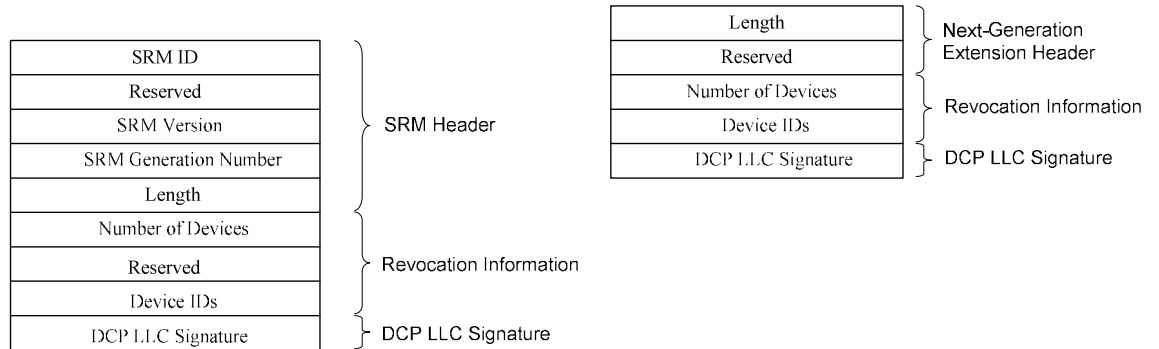
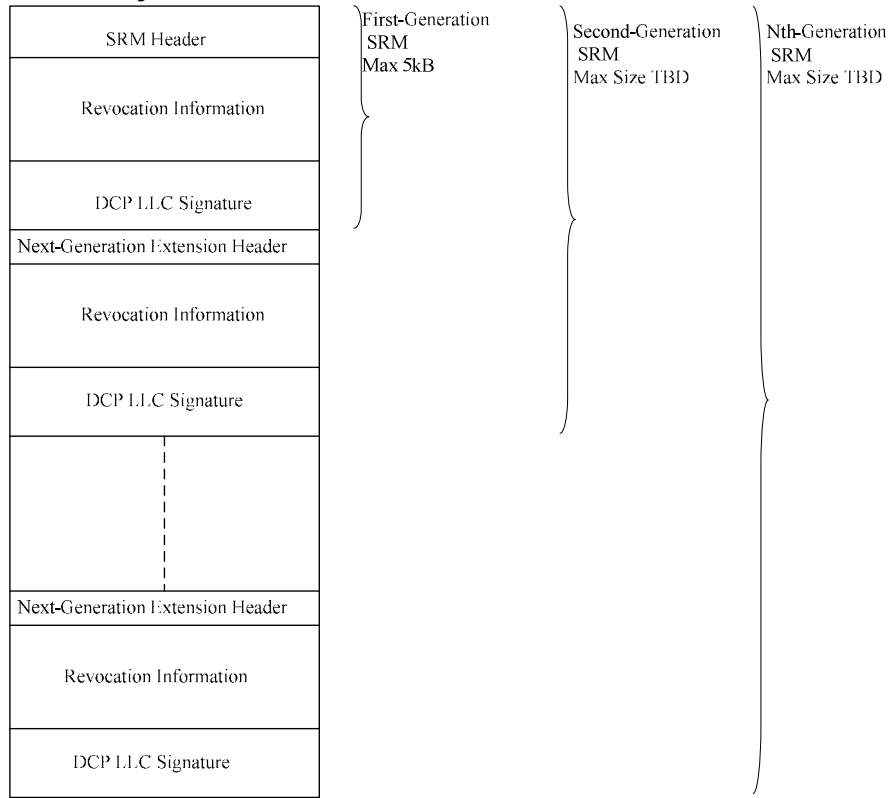
It is contemplated that an authorized participant in the authentication protocol may become compromised so as to expose the RSA private keys it possesses for misuse by unauthorized parties. In consideration of this, each HDCP Receiver is issued a unique Receiver ID which is contained in *cert<sub>r<sub>x</sub></sub>*. Through a process defined in the HDCP Adopter's License, the Digital Content Protection LLC may determine that an HDCP Receiver's RSA private key, *kpriv<sub>r<sub>x</sub></sub>*, has been compromised. If so, it places the corresponding Receiver ID on a revocation list that the HDCP Transmitter checks during authentication.

The HDCP Transmitter is required to manage system renewability messages (SRMs) carrying the Receiver ID revocation list. The validity of an SRM is established by verifying the integrity of its signature with the Digital Content Protection LLC public key, which is specified by the Digital Content Protection LLC.

For interoperability with HDCP 1.x, KSVs of revoked HDCP 1.x devices will be included in the HDCP 2 SRM, in addition to the HDCP 1.x SRM. Similarly, Receiver IDs of revoked HDCP 2 devices will be included in the HDCP 1.x SRM, in addition to the HDCP 2 SRM.

The SRMs are delivered with content and must be checked when available. The Receiver IDs must immediately be checked against the SRM when a new version of the SRM is received. Additionally, devices compliant with HDCP 2.0 and higher must be capable of storing at least 5kB of the SRM in their non-volatile memory. The process by which a device compliant with HDCP 2.0 or higher updates the SRM stored in its non-volatile storage when presented with a newer SRM version is explained in Section 5.2.

**5.1 SRM Size and Scalability**



**Figure 5.1. SRM Generational Format**

As illustrated in Figure 5.1, the size of the First-Generation HDCP SRM will be limited to a maximum of 5kB. The actual size of the First-Generation SRM is 5116 bytes. For scalability of the SRM, the SRM format supports next-generation extensions. By supporting generations of SRMs, an HDCP SRM can, if required in future, grow beyond the 5kB limit to accommodate more Receiver IDs. Next-generation extensions are appended to the current-generation SRM in order to ensure backward compatibility with devices that support only previous-generation SRMs.

Table 5.1 gives the format of the HDCP 2 SRM. All values are stored in big endian format.

Name	Size (bits)	Function
------	-------------	----------

SRM ID	4	A value of 0x9 signifies that the message is for HDCP2. All other values are reserved. SRMs with values other than 0x9 must be ignored.
HDCP2 Indicator	4	A value of 0x1 signifies that the message is for HDCP2
Reserved	8	Reserved for future definition. Must be 0x00
SRM Version	16	Sequentially increasing unique SRM numbers. Higher numbered SRMs are more recent
SRM Generation Number	8	Indicates the generation of the SRM. The generation number starts at 1 and increases sequentially
Length	24	Length in bytes and includes the combined size of this field (three bytes) and all following fields contained in the first-generation SRM i.e. size of this field, Number of Devices field, Reserved (22 bits) field, Device IDs field and Digital Content Protection LLC signature field (384 bytes) in the first-generation SRM
Number of Devices	10	Specifies the number (N1) of Receiver IDs / KSVs contained in the first-generation SRM
Reserved	22	Reserved for future definition. All bits set to 0
Device IDs	40 * N1 Max size for this field is 37760 (4720 bytes)	40-bit Receiver IDs / KSVs
DCP LLC Signature	3072	A cryptographic signature calculated over all preceding fields of the SRM. RSASSA-PKCS1-v1_5 is the signature scheme used as defined by PKCS #1 V2.1: RSA Cryptography Standard. SHA-256 is the underlying hash function

**Table 5.1. System Renewability Message Format**

Each subsequent next-generation extensions to the first-generation SRM will have the following fields.

Name	Size (bits)	Function
Length	16	Length in bytes and includes the combined size of this field (two bytes) and all following fields contained in this next-generation extension i.e. size of this field, Number of Devices field, Reserved (6 bits) field, Device IDs field and Digital Content Protection LLC signature field (384 bytes) in this next-generation SRM
Reserved	6	Reserved for future definition. All bits set to 0
Number of Devices	10	Specifies the number (N2) of Receiver IDs / KSVs contained in this next generation extension
Device IDs	40 * N2	40-bit Receiver IDs / KSVs
DCP LLC Signature	3072	A cryptographic signature calculated over all preceding fields of the SRM. RSASSA-PKCS1-v1_5 is the signature scheme used as defined by PKCS #1 V2.1: RSA Cryptography Standard. SHA-256 is the underlying hash function

**Table 5.2. Next-generation extension format**

## 5.2 Updating SRMs

The stored HDCP SRM must be updated when a newer version of the SRM is delivered with the content. The procedure for updating an SRM is as follows:

1. Verify that the version number of the new SRM is greater than the version number of the SRM currently stored in the device's non-volatile storage
2. If the version number of the new SRM is greater (implying that it is a more recent version), verify the signature on the new SRM

On successful signature verification, replace the current SRM in the device's non-volatile storage with the new SRM. If, for instance, the device supports only second-generation SRMs and the new SRM is a third-generation SRM, the device is not required to store the third-generation extension. Devices compliant with HDCP 2.0 or higher must be capable of storing at least 5kB (actual size is 5117 bytes) of the SRM (First-Generation SRM).

**Appendix A. Confidentiality and Integrity of Values**

Table A.1 identifies the requirements of confidentiality and integrity for values within the protocol. A *confidential* value must never be revealed. The *integrity* of many values in the system is protected by fail-safe mechanisms of the protocol. Values that are not protected in this manner require active measures beyond the protocol to ensure integrity. Such values are noted in the table as requiring integrity.

Value	Confidentiality Required <sup>±</sup> ?	Integrity Required <sup>±</sup> ?	Values Used by Core Functions?	Core Function
$lc_{128}$	Yes	Yes	Yes	HDCP Encryption and Decryption
$k_{pub_{dep}}$	No	Yes	No	
$cert_{rx}$	No	No	No	
$k_{pub_{rx}}$	No	Yes	No	
Receiver ID	No	Yes	No	
$k_{priv_{rx}}$	Yes	Yes	Yes	Handling Device Secret Key, during AKE, in plaintext form
$r_{tx}$	No	Yes*	Yes	
$r_{iv}$	No	Yes*	Yes	
REPEATER	No	Yes	No	
$r_{rx}$	No	Yes**	Yes	
$k_m$	Yes	Yes*	Yes	Handling of Master Key, during AKE (including Pairing) and Key Derivation, in plaintext form
$k_d$	Yes	Yes*	No	
$dkey_{0,1}$	Yes	Yes*	No	

<sup>±</sup> According to the robustness rules in the HDCP Adopter's License

\* Only within the transmitter

\* Only within the transmitter

\*\* Only within the receiver



Revision 2.0

Intel and DCP LLC Confidential

dkey <sub>2</sub>	Yes	Yes*	Yes	Handling of information or materials during Key Derivation and SKE, including but not limited to cryptographic keys used to encrypt or decrypt HDCP Core Keys ( $k_s$ ), from which HDCP Core Keys could reasonably be derived.
H	Yes	Yes	Yes	
H'	No	No	No	
m	No	No	Yes	
k <sub>h</sub>	Yes	Yes	Yes	Handling of information or materials during Pairing, including but not limited to cryptographic keys used to encrypt or decrypt HDCP Core Keys ( $k_m$ ), from which HDCP Core Keys could reasonably be derived.
r <sub>n</sub>	No	Yes*	Yes	
L	Yes	Yes	No	
L'	No	No	No	
k <sub>s</sub>	Yes	Yes*	Yes	Handling of Session Key, during SKE and HDCP Encryption/Decryption, in plaintext form
V[255:128]	Yes	Yes	No	
V'[127:0]	Yes	Yes	No	
V[127:0]	No	No	No	
V'[255:128]	No	No	No	
M	Yes	Yes	No	

Revision 2.0

Intel and DCP LLC Confidential

<i>M'</i>	No	No	No	
Receiver ID list	No	Yes	No	
DEPTH	No	Yes	No	
DEVICE_COUNT	No	Yes	No	
MAX_DEVS_EXCEEDED	No	Yes	No	
MAX_CASCADE_EXCEEDED	No	Yes	No	
<i>coarseInputCtr</i>	No	Yes*	Yes	HDCP Encryption and Decryption

**Table A.1. Confidentiality and Integrity of Values**

**Appendix B. DCP LLC Public Key**

Table B.1 gives the production DCP LLC public key.

Parameter	Value (hexadecimal)
Modulus n	B0E9 AA45 F129 BA0A 1CBE 1757 28EB 2B4E 8FD0 C06A AD79 980F 8D43 8D47 04B8 2BF4 1521 5619 0140 013B D091 9062 9E89 C227 8ECF B6DB CE3F 7210 5093 8C23 2983 7B80 64A7 59E8 6167 4CBC D858 B8F1 D4F8 2C37 9816 260E 4EF9 4EEE 24DE CCD1 4B4B C506 7AFB 4965 E6C0 0083 481E 8E42 2A53 A0F5 3729 2B5A F973 C59A A1B5 B574 7C06 DC7B 7CDC 6C6E 826B 4988 D41B 25E0 EED1 79BD 3985 FA4F 25EC 7019 23C1 B9A6 D97E 3EDA 48A9 58E3 1814 1E9F 307F 4CA8 AE53 2266 2BBE 24CB 4766 FC83 CF5C 2D1E 3AAB AB06 BE05 AA1A 9B2D B7A6 54F3 632B 97BF 93BE C1AF 2139 490C E931 90CC C2BB 3C02 C4E2 BDBD 2F84 639B D2DD 783E 90C6 C5AC 1677 2E69 6C77 FDED 8A4D 6A8C A3A9 256C 21FD B294 0C84 AA07 2926 46F7 9B3A 1987 E09F EB30 A8F5 64EB 07F1 E9DB F9AF 2C8B 697E 2E67 393F F3A6 E5CD DA24 9BA2 7872 F0A2 27C3 E025 B4A1 046A 5980 27B5 DAB4 B453 973B 2899 ACF4 9627 0F7F 300C 4AAF CB9E D871 2824 3EBC 3515 BE13 EBAF 4301 BD61 2454 349F 733E B510 9FC9 FC80 E84D E332 968F 8810 2325 F3D3 3E6E 6DBB DC29 66EB
Public Exponent e	03

**Table B.1. DCP LLC Public Key**