Video Services Forum (VSF)
Technical Recommendation TR-10-5

Internet Protocol Media Experience (IPMX): HDCP Key Exchange Protocol

February 23, 2024

INTELLECTUAL PROPERTY RIGHTS

RECIPIENTS OF THIS DOCUMENT ARE REQUESTED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT CLAIMS OR OTHER INTELLECTUAL PROPERTY RIGHTS OF WHICH THEY MAY BE AWARE THAT MIGHT BE INFRINGED BY ANY IMPLEMENTATION OF THE RECOMMENDATION SET FORTH IN THIS DOCUMENT, AND TO PROVIDE SUPPORTING DOCUMENTATION.

THIS RECOMMENDATION IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS RECOMMENDATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY MPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS RECOMMENDATION.

LIMITATION OF LIABILITY

VSF SHALL NOT BE LIABLE FOR ANY AND ALL DAMAGES, DIRECT OR INDIRECT, ARISING FROM OR RELATING TO ANY USE OF THE CONTENTS CONTAINED HEREIN, INCLUDING WITHOUT LIMITATION ANY AND ALL INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS, LOSS OF PROFITS, LITIGATION, OR THE LIKE), WHETHER BASED UPON BREACH OF CONTRACT, BREACH OF WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE FOREGOING NEGATION OF DAMAGES IS A FUNDAMENTAL ELEMENT OF THE USE OF THE CONTENTS HEREOF, AND THESE CONTENTS WOULD NOT BE PUBLISHED BY VSF WITHOUT SUCH LIMITATIONS.

Executive Summary

Internet Protocol Media Experience (IPMX) was created to foster the adoption of open standards-based protocols for interoperability over IP in the media and entertainment (M&E) and professional audio/video industries.

SMPTE ST 2110 and/or IPMX devices may be connected to baseband equipment which incorporates protection (specifically High-bandwidth Digital Content Protection System (HDCP)) to prevent the unauthorized copying content while in transit between sources and sinks. In this case, equipment which transports and streams this protected content must comply with HDCP standards.

This document describes interoperability requirements for devices implementing the "High-bandwidth Digital Content Protection System: Direct Adaptation Amendment, Revision 2.3, May 19, 2021" amendment of the "High-bandwidth Digital Content Protection System: Interface Independent Adaptation, Revision 2.3, May 02, 2018" specification.

Table of Contents

1 Introduction (Informative) 5
1.1 Contributors 6
1.2 About the Video Services Forum 6
2 Conformance Notation 7
3 Normative References 8
4 Acronyms 8
5 Definitions 9
6 A Sender in an HDCP System 11
6.1 Examples of Senders in an HDCP System (informative) 12
7 A Receiver in an HDCP System 12
7.1 Examples of Receivers in an HDCP System (informative) 13
8 A Repeater in an HDCP System 14
8.1 Examples of Repeaters in an HDCP System (informative) 15
9 HDCP System 15
9.1 HDCP on HDMI versus HDCP on RTP (informative) 17
9.2 Non-HDCP$_{RTP}$ v2.3 Connections 18
9.3 HDCP$_{RTP}$ v2.3 Connections 18
9.3.1 About Receivers 19
9.3.2 About Repeaters 19
9.3.3 About TX_{ip} and RX_{ip} Ports 19
9.3.4 About Protocol Layers (informative) 19
10 SDP Transport File 20
11 Constants 21
12 The HKEP Protocol 21
12.1 HKEP Overview (informative) 22
12.2 HKEP Container 22
12.3 HKEP Message Size 22
12.4 HKEP Messages 23
12.4.1 Protocol Version 23
12.4.2 AKE_PreInit message 24

12.4.3 AKE_PreInitStatus message 25
12.5 Vendor Extensions 26
12.6 The receiver protocol 26
12.6.1 With explicit pairing 27
12.6.2 With implicit pairing 28
12.6.3 With reconnect 30
12.6.4 Protocol illustration with explicit pairing (informative) 31
12.6.5 Protocol illustration with implicit pairing (informative) 32
12.6.6 Protocol illustration with reconnect (informative) 32
12.7 The non-receiver protocol 32
12.7.1 Protocol illustration for non-receivers (informative) 33
12.8 Sender (server side) protocol flowcharts (informative) 34
12.9 Receiver (client side) protocol flowcharts (informative) 35
12.10 Protocol sequence of a TCP/IP Connection (informative) 36
13 The $HDCP_{RTP}$ v2.3 protocol 38
13.1 Locality check 38
13.2 Authentication with repeaters 38
13.2.1 Exchange sequence 39
13.2.2 A first successful exchange 40
13.2.3 A subsequent successful exchange 40
13.2.4 Null Topology 41
13.2.5 Exchange sequence illustration (informative) 42
13.3 RepeaterAuth_Stream_Manage 42
14 $HDCP_{RTP}$ v2.3 media stream private data 43
14.1 Signaling the receivers 43
15 Encrypted HDCP Content (informative) 44

1 Introduction (Informative)

This VSF Technical Recommendation (TR) describes the HDCP Key Exchange Protocol (HKEP) for devices implementing the ST 2110 and/or IPMX and HDCP$_{RTP}$ v2.3 standards and technical recommendations.

The objective, at the system level, is to distribute HDCP$_{RTP}$ v2.3 encryption keys among a set of ST 2110/IPMX devices producing and consuming audio-video HDCP Content over multicast IP networks. The protocol aims at being as efficient and lightweight as possible, because IPMX devices are expected to vary greatly in terms of capabilities and processing power, and because we seek to have IPMX implemented in low cost devices.

The HKEP protocol is based on the "HDCP Direct Adaptation Amendment Rev. v2.3" amendment of the Digital Content Protection LLC (DCP LLC) specification "HDCP Interface Independent Adaptation Specification Revision 2.3". It requires ST 2110/IPMX devices to have an HDCP v2.2+ compliant public certificate issued by the DCP LLC consortium as a minimum.

The HKEP protocol defines the behaviors of ST 2110/IPMX devices with respect to optional features or behaviors of the HDCP$_{RTP}$ v2.3 specification. It also defines operational guidelines to obtain a high level of device interoperability. The HKEP specification does not change or replace any of the requirements or behaviors of the HDCP$_{RTP}$ v2.3 specification. Devices implementing the HKEP protocol are expected to behave according to the HDCP$_{RTP}$ v2.3 specification and be fully compliant with it.

The HKEP and HDCP protocol messages are exchanged through an out-of-band TCP/IP channel. The encrypted content is transmitted over RTP/UDP/IP channels as per the ST 2110 standard and IPMX specifications. These channels taken together behave as an HDCP-protected Interface Port as per the HDCP$_{RTP}$ v2.3 specification. The TCP/IP and RTP/UDP/IP channels can be on the same or on different network interfaces.

1.1 Contributors

The following individuals participated in the Video Services Forum IPMX working group that developed this technical recommendation.

Alain Bouchard (Matrox)
Danny Pierini (Matrox)
Wes Simpson (LearnIPVideo)
Jed Deame (Nextera)
Jean Lapierre (Matrox)
John Fletcher (BBC)
Paulo Francisco (Evertz AV)
Jean-Baptiste Lorent (IntoPIX)
Andreas Hildebrand (ALC NetworX)
Karl Johnson (Christie Digital)
Jack Douglas (PacketStorm)
Arnaud Germain (Intopix)
Ron Stites (Macnica)
Andrew Starks (Macnica)
Chris Lapp (Cisco)
Brad Gilmer (VSF)
Andre Testa (Matrox)
Greg Stigall (Warner Media)
Peter Brightwell (BBC)
Karl Paulsen (Diversified)
Clark Williams (Christie Digital)
Bob Ruhl (VSF)
Lynn Rowe (Consultant to Crown Castle)
Bob Baker (Clark)
Robert Welch (Arista)
Marc Levy (Macnica)
Scott Olsen (Warner Media)
Tadahiro Watanabe (Macnica)
Teiji Kubota (Macnica)
Daniel Bouquet (Analog Way)

1.2 About the Video Services Forum

The Video Services Forum, Inc. (www.videoservicesforum.org) is an international association dedicated to video transport technologies, interoperability, quality metrics and education. The VSF is composed of service providers, users and manufacturers. The organization's activities include:

- providing forums to identify issues involving the development, engineering, installation, testing and maintenance of audio and video services;

- exchanging non-proprietary information to promote the development of video transport service technology and to foster resolution of issues common to the video services industry;
- identification of video services applications and educational services utilizing video transport services;
- promoting interoperability and encouraging technical standards for national and international standards bodies.

The VSF is an association incorporated under the Not For Profit Corporation Law of the State of New York. Membership is open to businesses, public sector organizations and individuals worldwide. For more information on the Video Services Forum or this document, please call +1 929-279-1995 or e-mail opsmgr@videoservicesforum.org.

2 Conformance Notation

Normative text describes elements of the design that are indispensable or contain the conformance language keywords: "shall," "should," or "may."

Informative text is potentially helpful to the user but not indispensable and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except the Introduction and any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed to conform to the document and from which no deviation is permitted.

The keywords "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document.

The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

A conformant implementation according to this document is one that includes all mandatory provisions ("shall") and, if implemented, all recommended provisions ("should") as described. A conformant implementation need not implement optional provisions ("may") and need not implement them as described.

Unless otherwise specified, the order of precedence of the types of normative information in this document shall be as follows: Normative prose shall be the authoritative definition; Tables shall be next; followed by formal languages; then figures; and then any other language forms.

3 Normative References

HDCP Direct Adaptation Amendment Rev. 2.3: "High-bandwidth Digital Content Protection System: Direct Adaptation Amendment, Revision 2.3, May 19, 2021"

HDCP Interface Independent Adaptation Specification Revision 2.3: "High-bandwidth Digital Content Protection System: Interface Independent Adaptation, Revision 2.3, March 02, 2018"

HDCP 2.3 on HDMI Specification: "High-bandwidth Digital Content Protection System: Mapping HDCP to HDMI, Revision 2.3, February 28, 2018"

HDCP 2.3 on DisplayPort Specification: "High-bandwidth Digital Content Protection System: Mapping HDCP to DisplayPort, Revision 2.3, January 22, 2019"

HDCP 2.3 on HDBaseT Specification: "High-bandwidth Digital Content Protection System: Mapping HDCP to HDBaseT, Revision 2.3, July 26, 2019"

JT-NM TR-1001-1:2020, v1.1: "Joint Task Force on Networked Media, Technical Recommendation TR-1001-1:2020 v1.1, November 11, 2020, System Environment and Device Behaviors For SMPTE ST 2110 Media Nodes in Engineered Networks - Networks, Registration and Connection Management"

VSF_TR-10-1:2022: "VSF Technical Recommendation: Internet Protocol Media Experience (IPMX): System Timing and Definitions"

VSF_TR-10-8:2022: "VSF Technical Recommendation: Internet Protocol Media Experience (IPMX): NMOS"

AMWA IS-04 NMOS v1.3.1: "AMWA IS-04 NMOS Discovery and Registration Specification v1.3.1"

4 Acronyms

AMWA	Advanced Media Workflow Association
LRU	Least Recently Used
IPMX	Internet Protocol Media Experience
NMOS	Networked Media Open Specifications

HDCP	High bandwidth Digital Content Protection system
HKEP	HDCP Key Exchange Protocol.

5 Definitions

HDCP v2.2+	HDCP version greater than or equal to version 2.2
HDCP Content	As per "HDCP Interface Independent Adaptation Specification Revision 2.3"
HDCP Device	As per "HDCP Interface Independent Adaptation Specification Revision 2.3"
HDCP Device Key Set	A Device Key Set as per "HDCP Interface Independent Adaptation Specification Revision 2.3"
$HDCP_{HDMI}$	HDCP corresponding to media transmitted on an HDMI interface.
HDCP Protocol Descriptor	Protocol Descriptor value of an HDCP Device Key Set.
HDCP Receiver	As per "HDCP Interface Independent Adaptation Specification Revision 2.3"
HDCP Receiver ID	As per "HDCP Interface Independent Adaptation Specification Revision 2.3"
HDCP Repeater	As per "HDCP Interface Independent Adaptation Specification Revision 2.3"
$HDCP_{RTP}$ v2.3	HDCP corresponding to media encapsulated in RTP as described in the "HDCP Direct Adaptation Amendment Rev. 2.3" specification of the DCP LLC "HDCP Interface Independent Adaptation Specification Revision 2.3."
HDCP System	As per "HDCP Interface Independent Adaptation Specification Revision 2.3"
HDCP Transmitter	As per "HDCP Interface Independent Adaptation Specification Revision 2.3"
NodeId	A globally unique identifier associated with a Sender that corresponds to the AMWA IS-04 NMOS node_id

	attribute of the Sender in an NMOS environment or to any other globally unique identifier in other environments. In both cases the identifier remains the same for all time (much in the same way as a serial number).
Null Topology	A topology described by an $HDCP_{RTP}$ v2.3 RepeaterAuth_Send_ReceiverID_List message having MAX_DEVS_EXCEEDED and MAX_CASCADE_EXCEEDED set to false, DEVICE_COUNT and DEPTH set to 0. Having DEPTH equal to 0 for an HDCP Repeater serves as an indication that the REPEATER is no longer subscribing to HDCP Content and is no longer considered part of the topology tree.
PortId	A locally unique identifier associated with a group of HDCP Content streams produced by a Sender.
ProtocolMaxContentStreams	An integer indicating the maximum number of content streams that can be declared in an $HDCP_{RTP}$ v2.3 RepeaterAuth_Stream_Manage message.
ProtocolVersionMajor	An integer in the range from 1 to 15 inclusively indicating the major version of the HKEP protocol.
ProtocolVersionMinor	An integer in the range from 0 to 15 inclusively indicating the minor version of the HKEP protocol.
ProtocolTimeout	An integer expressed in milliseconds indicating the maximum amount of time allowed for a protocol operation to complete successfully.
Receiver	As per "JT-NM TR-1001-1:2020, v1.1", it is a "Receiver Media Node" that consumes ST 2110/IPMX media streams.
Repeater	A "Media Node" that is both a Sender and a Receiver. It is described as having a Receiver side and a Sender side. Unless otherwise specified the Receiver side is referenced as a Receiver and the Sender side as a Sender.
RX Port	An HDCP Receiver as per "HDCP 2.3 on HDMI Specification", "HDCP 2.3 on DisplayPort Specification", "HDCP 2.3 on HDBaseT Specification" or any other adaptation that is not based on the "HDCP Direct Adaptation Amendment Rev. 2.3" specification.

RX_{ip} Port	An HDCP Receiver that uses RTP/UDP/IP protocol to encapsulate the media and TCP/IP to transport messages as per "HDCP Direct Adaptation Amendment Rev. 2.3" specification of the DCP LLC "HDCP Interface Independent Adaptation Specification Revision 2.3."
Self-Subscribing HDCP Repeater	An HDCP Repeater that virtually subscribes to itself as an active downstream HDCP Receiver when no other active downstream HDCP Devices are connected or authenticated.
Sender	As per "JT-NM TR-1001-1:2020, v1.1", it is a "Sender Media Node" that produces ST 2110/IPMX media streams.
ST 2110	SMPTE 2110 is a suite of standards from the Society of Motion Picture and Television Engineers (SMPTE) that describes how to send digital media over an IP network.
TX Port	An HDCP Transmitter as per "HDCP 2.3 on HDMI Specification", "HDCP 2.3 on DisplayPort Specification", "HDCP 2.3 on HDBaseT Specification" or any other adaptation that is not based on the "HDCP Direct Adaptation Amendment Rev. 2.3" specification.
TX_{ip} Port	An HDCP Transmitter that uses RTP/UDP/IP protocol to encapsulate the media and TCP/IP to transport messages as per "HDCP Direct Adaptation Amendment Rev. 2.3" specification of the DCP LLC "HDCP Interface Independent Adaptation Specification Revision 2.3."

6 A Sender in an HDCP System

A Sender in an IP-based HDCP System is either an HDCP Transmitter or an HDCP Repeater. As an HDCP Repeater it shall have at least one TX_{ip} Port and it shall have exactly one RX Port and no RX_{ip} Port. It may have any number of TX Ports. As an HDCP Transmitter it shall have exactly one TX_{ip} Port.

A Sender participating in an HDCP System shall produce media streams compliant with $HDCP_{RTP}$ v2.3 and the normative text of this technical recommendation. The Sender shall behave in compliance with $HDCP_{RTP}$ v2.3 and the normative text of this technical recommendation.

A Sender shall have an associated NodeId.

A Sender shall use the same HDCP session key k_s and initial vector r_{iv} for all its encrypted HDCP Content streams.

Note: A physical device may have connectors and ports that are not part of the description of a Sender in an HDCP System (see section 9 for more details on HDCP Systems). This is because this specification is interested only in the ports that connect the devices in an HDCP System topology tree.

6.1 Examples of Senders in an HDCP System (informative)

The first row of Figure 1 illustrates Senders as HDCP Repeaters while the second row illustrates them as HDCP Transmitters.

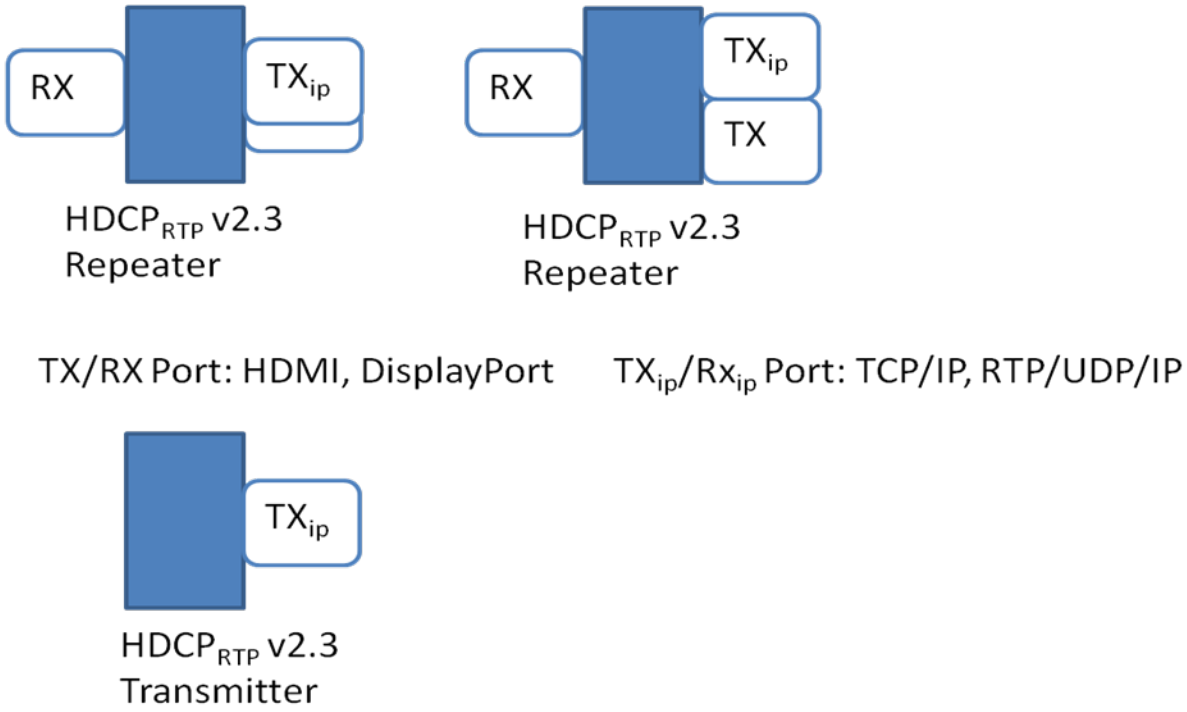

Figure 1- Examples of Senders

7 A Receiver in an HDCP System

A Receiver in an IP-based HDCP System is an HDCP Repeater since HKEP does not support true HDCP Receivers. It shall have at least one RX_{ip} Port and it shall have no TX_{ip} Port. It may have any number of TX Ports.

A Receiver participating in an HDCP System shall consume media streams compliant with $HDCP_{RTP}$ v2.3 and the normative text of this technical recommendation. The Receiver shall behave in compliance with $HDCP_{RTP}$ v2.3 and the normative text of this technical recommendation.

A Receiver subscribing to a Sender's HDCP Content shall behave as a Self-Subscribing HDCP Repeater unless at least one active downstream HDCP Device is connected and authenticated to prevent a transition from C5 to C0 as shown in "Figure 2.17. HDCP Repeater Upstream Authentication Protocol State Diagram" of the $HDCP_{RTP}$ v2.3 protocol.

A Receiver shall unsubscribe from a Sender's HDCP Content when no active downstream HDCP Devices are connected or authenticated, unless it is acting as a Self-Subscribing HDCP Repeater.

Note: This implies that a Receiver cannot appear in an HDCP System as an HDCP Receiver. It will always appear as an HDCP Repeater, possibly as a Self-Subscribing HDCP Repeater or as an HDCP Repeater having an internal active downstream HDCP Receiver. As such, a Receiver without any physical output connector cannot count for less than 2 HDCP Devices in an HDCP System topology tree (the RX_{ip} Port and either an internal Self-Subscribing RX_{ip} Port or an internal RX Port). This restriction relates to the normative text of section 2.10 of the HDCP Interface Independent Adaptation Specification Revision 2.3 which states, "If an HDCP Repeater has no active downstream HDCP Devices, it must authenticate as an HDCP Receiver with REPEATER set to ‘false’ if it wishes to receive HDCP Content, but must not pass HDCP Content to downstream devices." In the case of HKEP, if an HDCP Repeater has no real active downstream HDCP Devices and it wishes to receive HDCP Content, it must authenticate as an HDCP Repeater with REPEATER set to 'true' and declare a virtual active downstream HDCP Device (Self-Subscribing).

A Receiver shall have an associated HDCP Receiver ID.

Note: A physical device can have connectors and ports that are not part of the description of a Receiver in an HDCP System (see section 9 for more details on HDCP Systems) because this specification is interested only in the ports that connect the devices in an HDCP System topology tree.

7.1 Examples of Receivers in an HDCP System (informative)

In Figure 2 the yellow internal RX_{ip} Port is the Self-Subscribing virtual downstream HDCP Receiver and the white internal RX Port represent an internal HDCP Receiver.

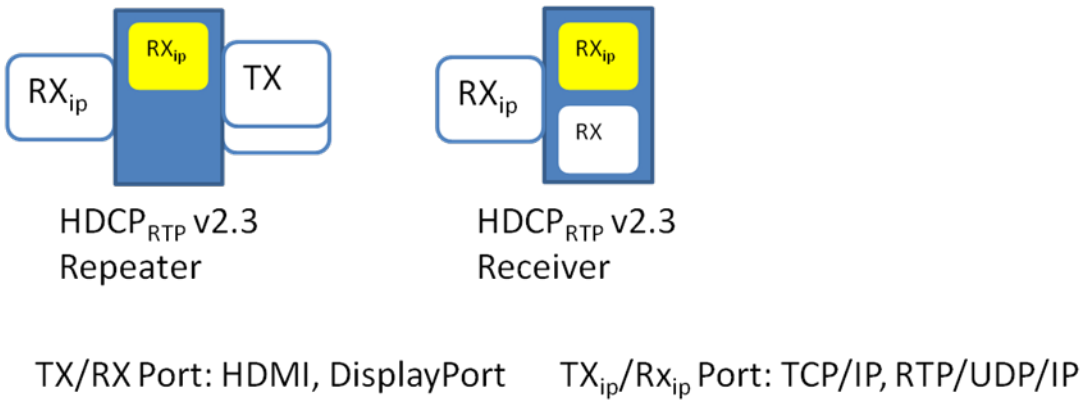

Figure 2 - Examples of Receivers

8 A Repeater in an HDCP System

A Repeater in an IP-based HDCP System is an HDCP Repeater. It shall have at least one TX_{ip} Port and it shall have exactly one RX_{ip} Port and no RX Port. It may have any number of TX Ports.

A Repeater participating in an HDCP System shall consume and produce media streams compliant with $HDCP_{RTP}$ v2.3 and the normative text of this technical recommendation. The Repeater shall behave in compliance with $HDCP_{RTP}$ v2.3 and the normative text of this technical recommendation.

The behavior of a Repeater is described either as a Repeater when considering the whole device, a Receiver when considering only the RX_{ip} side, or a Sender when considering only the TX_{ip} side. In this technical recommendation, unless otherwise specified, a Receiver refers to either the Receiver side of a Repeater, or a simple Receiver. Similarly, unless otherwise specified, a Sender refers to either the Sender side of a Repeater, or a simple Sender.

A Repeater subscribing to a Sender's HDCP Content shall behave as a Self-Subscribing HDCP Repeater unless at least one active downstream HDCP Device is connected and authenticated in order to prevent a transition from C5 to C0 as shown in "Figure 2.17. HDCP Repeater Upstream Authentication Protocol State Diagram" of the $HDCP_{RTP}$ v2.3 protocol.

A Receiver shall unsubscribe from a Sender's HDCP Content when no active downstream HDCP Devices are connected or authenticated, unless it behaves as a Self-Subscribing HDCP Repeater.

A Repeater shall have an associated NodeId.

A Repeater shall use the same HDCP session key k_s and initial vector r_{iv} for all its encrypted HDCP Content streams.

Note: A physical device can have connectors and ports that are not part of the description of a Repeater in an HDCP System (see section 9 for more details on HDCP Systems) because this specification is interested only in the ports that connect the devices in an HDCP System topology tree.

8.1 Examples of Repeaters in an HDCP System (informative)

In Figure 3 the yellow internal RX_{ip} Port is the Self-Subscribing virtual downstream HDCP Receiver.

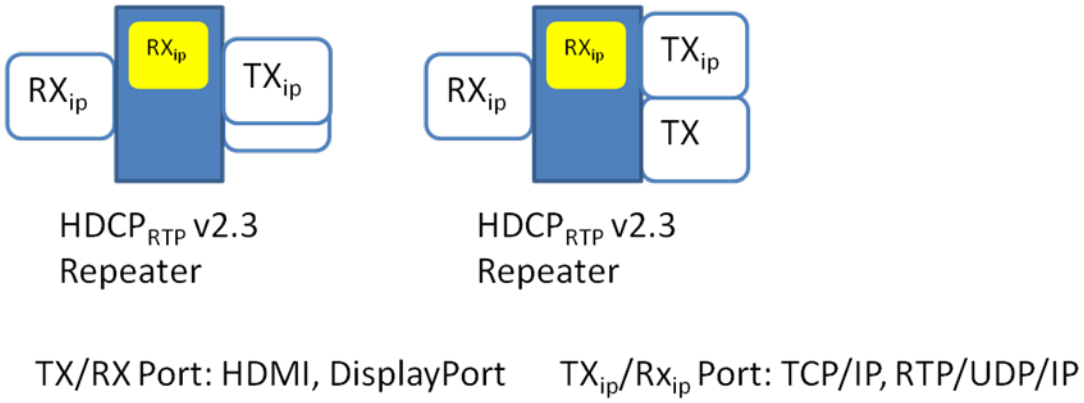

Figure 3 - Examples of Repeaters

9 HDCP System

An HDCP System represents a topology tree of devices through which flow HDCP Content. At the root of the tree there is an HDCP Transmitter (the most upstream device). In between the root and the leaves there are a number of optional layers of HDCP Repeaters. Finally, at the leaves of the tree there are a non-zero number of HDCP Receivers (the most downstream devices).

HDCP Content flows from the HDCP Transmitter at the root of the topology tree to either an HDCP Repeater (receiver side) or an HDCP Receiver. HDCP Content flows from an HDCP Repeater (sender side) to another HDCP Repeater (receiver side) or an HDCP Receiver.

HDCP Content flows through the HDCP System through RX Ports, TX Ports, RX_{ip} Ports and TX_{ip} Ports.

The group of streams making the HDCP Content that originates from the TX_{ip} Port of a Sender acting as an HDCP Transmitter at the root of the topology tree, shall have an associated PortId value. It is such a group of streams, or a subset of such a group of streams, that flow through the HDCP System.

Similarly, the group of streams making the HDCP Content that originates from the TX Port of an HDCP device, acting as an HDCP Transmitter at the root of the topology tree, shall have an

associated PortId value. It is such a group of streams, or a subset of such a group of streams, that flow through the HDCP System.

The topology tree of an HDCP System describes the devices through which flow all or a subset of the HDCP Content streams originating from the HDCP Transmitter at the root of the topology tree. Each Sender or Repeater in a topology tree shall associate a PortId value to such a group or sub-group of HDCP Content streams.

The PortId value shall be locally unique for a given Sender or Repeater identified by a globally unique NodeId value. The PortId value identifies the HDCP Content flowing through an HDCP System.

Note: For example, an HDCP Transmitter having an HDMI output connector transmitting HDCP Content can carry multiple audio/video streams. Such a group of HDCP Content streams is associated with a PortId value. The concept of PortId is straightforward for HDMI and DisplayPort interfaces as it identifies all the content streams transmitted through a given physical connector. For an ST 2110/IPMX interface the concept of PortId does not refer to a physical interface but to the HDCP Content flowing through an ST 2110/IPMX interface in an HDCP System.

Note: A Sender, Repeater or Receiver may simultaneously participate in multiple topology trees. For example, a Sender acting as an HDCP Repeater having two physical HDMI input connectors (A and B) can participate in two HDCP Systems: a first having the HDCP Transmitter connected to the physical HDMI connector A as the root of the topology tree and a second having the HDCP Transmitter connected to the physical HDMI connector B as the root of the topology tree. Each HDCP System operates independently of the others. The HDCP Content streams flowing through the HDMI connector A must be identified independently of the HDCP Content streams flowing through the HDMI connector B.

Note: A second example of this is a Receiver acting as an HDCP Repeater with an internal HDCP Receiver doing picture-in-picture which can participate in two HDCP Systems: a first having a Sender identified by NodeId X which HDCP Content flows as the main picture on the screen, and a second having a Sender identified by NodeId Y which HDCP Content flows as the sub-picture on the screen. Each HDCP System operates independently of the others. The HDCP Content streams flowing through the Receiver as the main picture must be identified independently of the content streams flowing through the Receiver as the sub-picture.

Note: A Repeater or Receiver participating in a topology tree gets the PortId value of the HDCP Content produced by a Sender from the port-id parameter of the a=hkep: session attribute of the SDP transport file associated with the subscribed ST 2110/IPMX media stream. It is locally unique on the Sender but it may not be locally unique on the Repeater which generates a new locally unique PortId for the HDCP Content streams it produces.

9.1 HDCP on HDMI versus HDCP on RTP (informative)

As illustrated in Figure 4, with HDCP 2.3 on HDMI ($HDCP_{HDMI}$) the HDMI cable provides two independent channels: the TMDS (transition-minimized differential signaling) unidirectional media channel to transport the content, and the I^2C channel to transport the status and control messages.

With $HDCP_{RTP}$ v2.3 there is a similar model where RTP/UDP/IP provides a high-speed unidirectional multicast media channel to transport the content, and TCP/IP provides a reliable channel to transport the status and control messages.

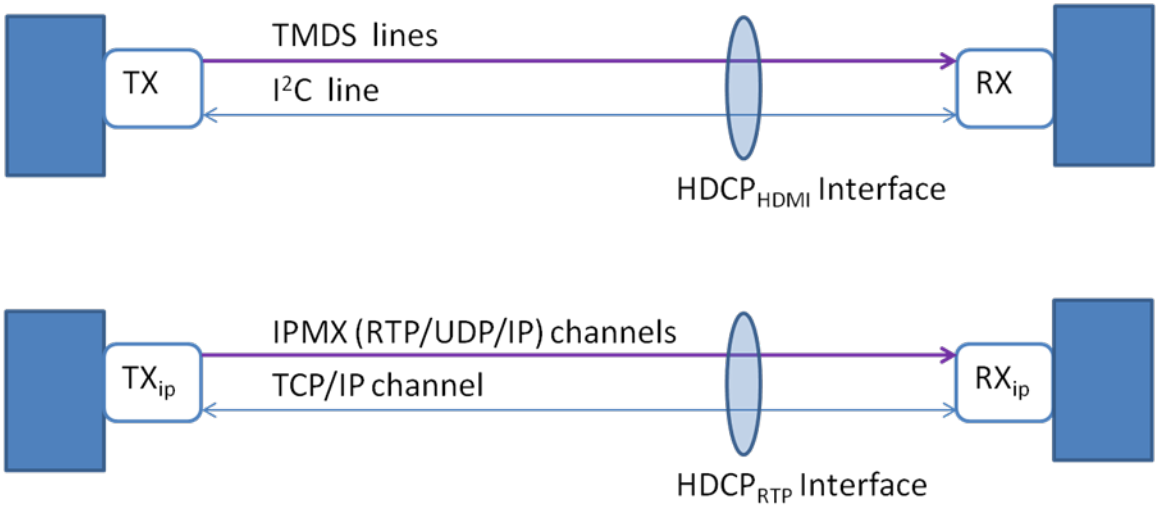

Figure 4 - HDCP on HDMI versus HDCP on RTP

A major distinction between the two systems is that an HDMI cable permanently and persistently connects a TX Port's physical HDMI output connector to an RX Port's physical HDMI input connector. It is a peer-to-peer connection where both the transport of status and control messages and media content are interrupted when the cable is disconnected. There may be a live signal on the TX Port but it is only when an HDMI cable is connected from the RX Port to the TX Port that media content starts to flow between the two ports. Similarly, it is only when an HDMI cable is connected from the RX Port to the TX Port that status and control messages start to flow between the two ports.

In an HDCP on RTP system the TCP/IP connection between the TX_{ip} Port and RX_{ip} Port is neither permanent nor persistent, and the multicast RTP/UDP/IP media channels represent a one-to-many distribution system. A disconnect of the TCP/IP channel does not interrupt the transport of media channels. A Receiver unsubscribing from a media channel does not interrupt the transport of media content to other subscribers or the transport of status and control messages. There may be a live signal on a media channel's multicast address but it is only when a Receiver subscribes to the media channel's multicast address that media content starts to flow between the

two ports. Similarly, it is only when a Receiver connects to a Sender through TCP/IP that status and control messages start to flow between the two ports.

Although the most generic model is described with multicast RTP/UDP/IP media channels, unicast RTP/UDP/IP media channels are also possible.

The multicast/unicast RTP/UDP/IP media channels may involve redundancy channels and/or error correction channels or any other mechanisms associated with the transport of the media content.

9.2 Non-HDCP$_{RTP}$ v2.3 Connections

The connection between an RX Port and a TX Port in a non-IP based HDCP System shall behave according to an HDCP interface specific specification such as the "HDCP 2.3 on HDMI Specification", the "HDCP 2.3 on DisplayPort Specification", the "HDCP 2.3 on HDBaseT Specification" or any other HDCP interface specific specification that is not HDCP$_{RTP}$ v2.3.

Note: For example, the "HDCP 2.3 on HDMI" protocol is used between a TX Port and an RX Port, each corresponding respectively to physical HDMI output and input connectors.

9.3 HDCP$_{RTP}$ v2.3 Connections

The connection between an RX$_{ip}$ Port and a TX$_{ip}$ Port in an IP-based HDCP System shall behave according to the HDCP$_{RTP}$ v2.3 interface specific specification and the normative text of this technical recommendation.

The HDCP Device Key Set associated with an RX$_{ip}$ Port shall have an HDCP Protocol Descriptor value equal to 0x01 to indicate an HDCP v2.2+ compliant device.

A Receiver subscribing to HDCP Content streams associated with a given PortId from a Sender shall create a TCP/IP connection from an RX$_{ip}$ Port (the client) of the Receiver and a TX$_{ip}$ Port (the server) of the Sender. After the TCP/IP "connect" operation, the client and server shall execute the HKEP protocol defined in this technical recommendation. The execution of the HKEP protocol results in the exchange of HDCP$_{RTP}$ v2.3 messages. After a successful execution of the HDCP$_{RTP}$ v2.3 protocol, the Receiver obtains the encryption key used by the Sender to encrypt the subscribed HDCP Content streams.

The HDCP$_{RTP}$ v2.3 protocol is used between a TX$_{ip}$ Port and an RX$_{ip}$ Port to distribute encryption keys and exchange HDCP$_{RTP}$ v2.3 messages.

Note: The TCP/IP connection is used to exchange HKEP and HDCP$_{RTP}$ v2.3 messages, while the encrypted HDCP Content is transmitted by a Sender to a number of Receivers using multicast RTP/UDP/IP according to the ST 2110/IPMX protocol.

Note: The TCP/IP and RTP/UDP/IP channels may reside on the same or on different network interfaces of a Sender or Receiver device.

9.3.1 About Receivers

A Receiver may subscribe to HDCP Content from a number of Senders, thus participating simultaneously in multiple HDCP Systems. All the RX_{ip} Ports of a Receiver shall share the same HDCP Device Key set, hence the same HDCP Receiver ID. As per the $HDCP_{RTP}$ v2.3 specification "HDCP Receivers and HDCP Repeaters with multiple inputs may share the same Public Key Certificates and Private Keys across all inputs".

9.3.2 About Repeaters

A Repeater may subscribe to HDCP Content from a number of Senders thus participating simultaneously in multiple HDCP Systems. All the RX_{ip} Ports of a Repeater shall share the same HDCP Device Key set, hence the same HDCP Receiver ID. As per the $HDCP_{RTP}$ v2.3 specification "HDCP Receivers and HDCP Repeaters with multiple inputs may share the same Public Key Certificates and Private Keys across all inputs".

9.3.3 About TX_{ip} and RX_{ip} Ports

The TX_{ip} Port of a Sender is spawned from a TCP/IP server socket associated with a given server address-port pair (server IP address, server port number). Each TCP/IP connection from a client spawns a new TX_{ip} Port corresponding to a socket associated with a new server address-port pair (server IP address, dynamic server port number). Multiple instances materialize when clients connect to the server and spawn multiple TX_{ip} Ports.

The RX_{ip} Port of a Receiver corresponds to a TCP/IP client socket associated with a given client address-port pair (client IP address, client port number). There are as many RX_{ip} Ports on a Receiver as there are connections to different Senders but because of the properties of the HDCP System, only one such RX_{ip} Port is used in a given topology tree. A Receiver shall use a single TCP/IP connection for all the HDCP Content streams produced by a given Sender.

9.3.4 About Protocol Layers (informative)

Figure 5 illustrates the various protocol layers of the system.

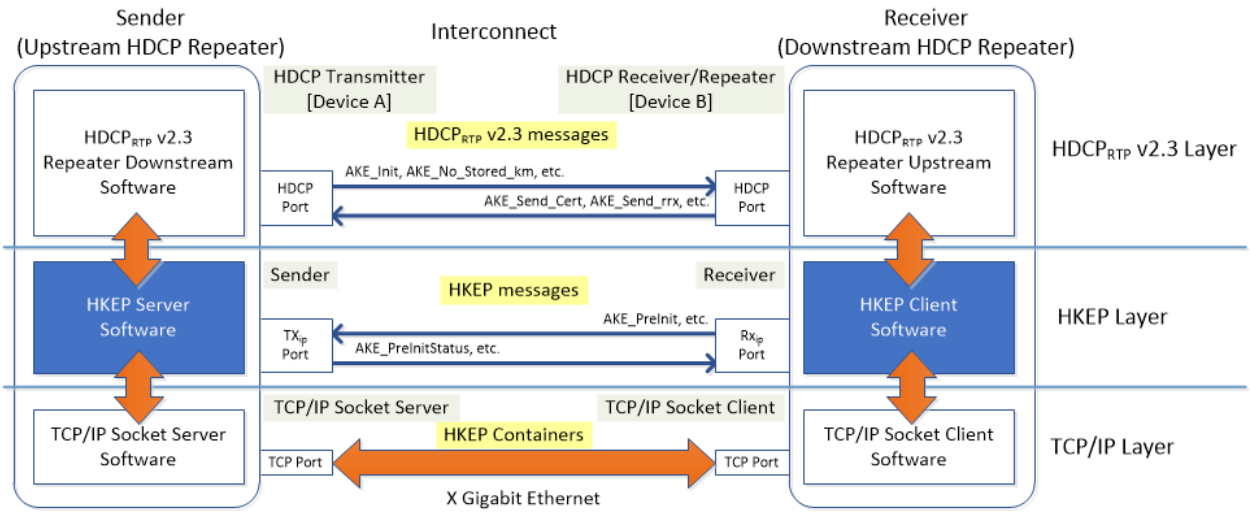

Figure 5 - Protocol Layers

- The HKEP software receives an HKEP container transferred from its lower layer TCP/IP socket software.
- The HKEP software gets the protocol message from the HKEP container and checks the msg_id value. If it is in the range from 1 to 31 inclusively, the message is an HDCP$_{RTP}$ v2.3 protocol message, otherwise if it is in the range from 32 to 63 inclusively the message is an HKEP protocol message, otherwise the message is invalid.
- If the message is an HKEP protocol message, the HKEP software processes it. Otherwise, if the message is an HDCP$_{RTP}$ v2.3 protocol message the HKEP software forwards it to the HDCP$_{RTP}$ v2.3 software for processing.

10 SDP Transport File

A Sender shall provide an SDP transport file for signalling the parameters of an ST 2110/IPMX media stream.

In an NMOS environment, a Sender shall make an associated SDP transport file available through the method described in VSF_TR-10-8.

The SDP transport file shall contain at least one "hkep" session attribute when the associated ST 2110/IPMX media stream is HDCP Content. The format is as follows:

```
a=hkep:<port> <nettype> <addrtype> <unicast-address> <node-id> <port-id>
```

<port> shall be the server port number of the Sender TCP/IP socket listening for HKEP connections.

<nettype> shall be IN

<addrtype> shall be either IP4 or IP6

<unicast-address>	shall be the IPv4 or IPv6 address of the Sender TCP/IP socket listening for HKEP connections.
<node-id>	shall be the NodeId value of the Sender producing the HDCP Content, formatted as a sequence of 32 hexadecimal digits formatted as xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
<port-id>	shall be the PortId value associated with the HDCP Content produced by the Sender, formatted as a sequence of 10 hexadecimal digits formatted as xx-xx-xx-xx-xx

The SDP transport file may contain multiple "hkep" session attributes, listed in order of preference from the point of view of the Sender.

A Receiver shall try the first item of the list and fallback to the next one if it cannot connect using the first one and so on.

Upon a successful connection with the Sender using the parameters from an entry of the list, a Receiver shall continue using the same parameters until a connection attempt fails, in which case the Receiver shall restart from the first entry of the list.

11 Constants

In this technical recommendation the following definitions shall have the following values

ProtocolVersionMajor	1
ProtocolVersionMinor	0
ProtocolTimeout	7000 milliseconds
ProtocolMaxContentStreams	128

12 The HKEP Protocol

The HKEP protocol describes a preamble to the $HDCP_{RTP}$ v2.3 protocol and prescribes the expected behavior of optional aspects of the $HDCP_{RTP}$ v2.3 protocol. It also describes the interactions of Senders and Receivers implementing the $HDCP_{RTP}$ v2.3 protocol.

The HKEP protocol shall transport the $HDCP_{RTP}$ v2.3 protocol messages over TCP/IP. The HKEP protocol layer is made of the AKE_PreInit and AKE_PreInitStatus messages and the encapsulation of the $HDCP_{RTP}$ v2.3 protocol messages.

12.1 HKEP Overview (informative)

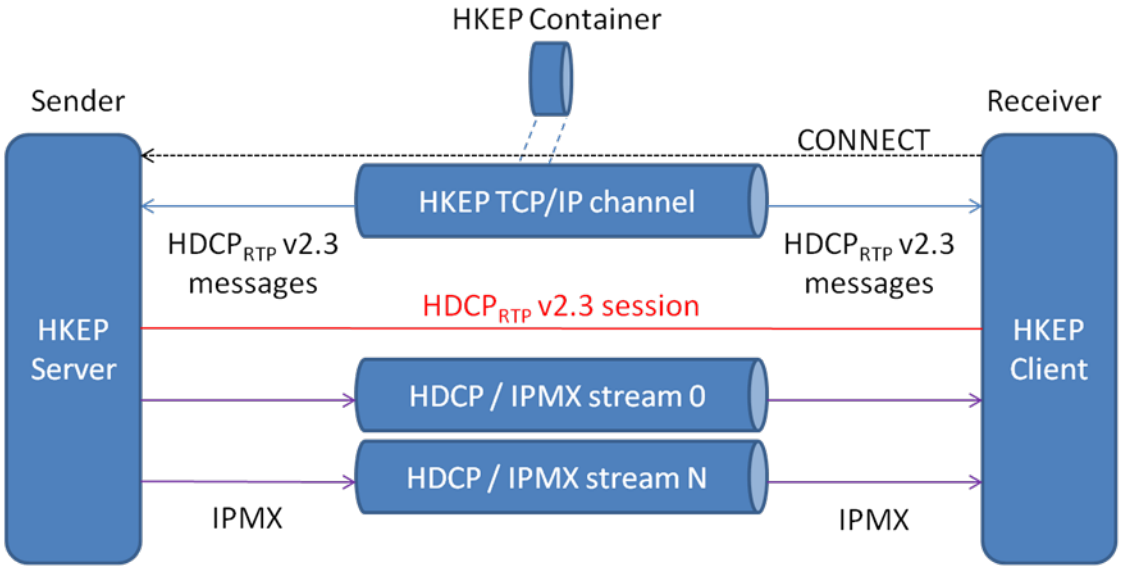

Figure 6 - HKEP Overview

12.2 HKEP Container

An HKEP container corresponds to the aggregation of { msgSize, HDCP$_{RTP}$ v2.3 message or HKEP message, a number of msgExtraBytes }. An "HDCP$_{RTP}$ v2.3 message" refers to an HDCP$_{RTP}$ v2.3 protocol message only, and an “HKEP message” refers to an HKEP protocol message only. An “augmented HDCP$_{RTP}$ v2.3 message” refers to the aggregation of { msgSize, HDCP$_{RTP}$ v2.3 message }, and an "augmented HKEP message" refers to the aggregation of { msgSize, HKEP message }.

Layer	Payload
HKEP	msgSize
HDCP / HKEP	HDCP$_{RTP}$ v2.3 message or HKEP message
HKEP	a number of msgExtraBytes

12.3 HKEP Message Size

All the HDCP$_{RTP}$ v2.3 protocol messages are augmented by a 2-byte unsigned integer (uint) prefix msg_size, indicating the total size of the HKEP container, including msg_size. The msg_size value shall be larger or equal to the sum of the sizes of the augmented HDCP$_{RTP}$ v2.3 message and msg_size. The message shall be decoded according to the HDCP$_{RTP}$ v2.3 specification of the message, and the extra bytes shall be ignored unless otherwise specified.

Similarly, all the HKEP protocol messages shall be augmented by a 2-byte uint prefix msg_size, indicating the total size of the HKEP container, including msg_size. The msg_size value shall be greater than or equal to the sum of the sizes of the augmented HKEP message and msg_size. The message shall be decoded according to the HKEP specification of the message, and the extra bytes shall be ignored unless otherwise specified.

The msg_size value shall be in big-endian format.

Note: A typical implementation would perform a message receive operation by first reading the big-endian msg_size from the TCP/IP stream and then read (msg_size - 2) bytes to get the remaining part of the augmented $HDCP_{RTP}$ v2.3 or the augmented HKEP message and possibly trailing extra bytes. The known elements of the message are processed and the trailing extra bytes are ignored.

Note: The msg_size prefix provides the necessary information to delimit the messages of the TCP/IP stream. Without such information full parsing of the messages is required in order to properly delimit them. Using a size prefix is a common approach of delimiting messages of a TCP/IP stream.

12.4 HKEP Messages

The HKEP protocol layer uses two msg_id values (message types) outside the valid range from 1 to 20 inclusively and the reserved range from 21 to 31 defined by the $HDCP_{RTP}$ v2.3 protocol. There is no interference with the $HDCP_{RTP}$ v2.3 protocol because HKEP messages are exchanged only during the preamble of the $HDCP_{RTP}$ v2.3 protocol.

The HKEP protocol defines the following messages:

MessageType	msg_id value
AKE_PreInit	32
AKE_PreInitStatus	33

12.4.1 Protocol Version

The version_major:version_minor attribute of the HKEP messages encodes the HKEP protocol version. The version is encoded into a byte with the version_major number in the 4 most significant bits and the version_minor number in the 4 least significant bits. The version_major and version_minor values shall be the ProtocolVersionMajor and ProtocolVersionMinor values defined in this document.

Both the client and server sides of a TCP/IP connection shall use the same HKEP protocol version. The server side may select a protocol version matching the protocol version of the client, as defined in the AKE_PreInit message. If the server side cannot match the client protocol version, it shall fail and respond with an AKE_PreInuitStatus message with the version_major:version_minor attribute set to the highest version supported by the server and the status attribute set to statusInvalidParameters. The client may retry to execute the HKEP protocol using the version from the AKE_PreInuitStatus response. If the server side can match the client protocol version, it shall respond with an AKE_PreInuitStatus message with the version_major:version_minor attribute of the client AKE_PreInit message.

Note: All the versions of the HKEP protocol will have as the first two bytes of the messages: msg_id and version_major :version_minor. Newer versions of the HKEP protocol may add new

messages, add new attributes at the end of actual messages or change the structure of actual messages with the exception of the first two bytes that are the same for all versions of the protocol.

12.4.2 AKE_PreInit message

All values shall be in big-endian format as per the $HDCP_{RTP}$ v2.3 specification.

Syntax	No. of Bytes	Identifier
AKE_PreInit {		
msg_id	1	uint
version_major:version_minor	1	uint
pairing	1	bool
restart / REAUTH_REQ	1	bool
receiver	1	bool
receiverId	5	uint
portId	5	uint
nodeId	16	uint
vendorExtension	16	uint
}		

12.4.2.1 msg_id

The msg_id attribute shall be AKE_PreInit (value 32).

12.4.2.2 version_major:version_minor

The version_major:version_minor shall indicate the protocol version as described at section 12.4.1.

12.4.2.3 pairing

The pairing flag shall be used to request an explicit pairing operation as described at section 12.6.1.

12.4.2.4 restart / REAUTH_REQ

The restart / REAUTH_REQ flag shall be used to request that the $HDCP_{RTP}$ v2.3 protocol start at the AKE phase.

12.4.2.5 receiver

The receiver flag shall be used to indicate that the client is a Receiver.

12.4.2.6 receiverId

The receiverId attribute shall indicate the HDCP Receiver ID of the client.

12.4.2.7 portId

The portId attribute shall indicate the PortId of the HDCP Content.

12.4.2.8 nodeId

The nodeId attribute shall indicate the NodeId of the Sender.

12.4.2.9 vendorExtension

The vendorExtension attribute may contain vendor specific payload as described at section 12.5.

12.4.3 AKE_PreInitStatus message

All values shall be in big-endian format as per the $HDCP_{RTP}$ v2.3 specification.

Syntax	No. of Bytes	Identifier
AKE_PreInitStatus {		
msg_id	1	uint
version_major:version_minor	1	uint
status	1	uint
pairingSlots	2	uint
sessionSlots	2	uint
vendorExtension	16	uint
}		

12.4.3.1 msg_id

The msg_id attribute shall be AKE_PreInitStatus (value 33).

12.4.3.2 version_major :version_minor

The version_major:version_minor attribute shall indicate the protocol version as described at section 12.4.1.

12.4.3.3 status

The status attribute shall indicate the state of a request and how to proceed after the reception of the AKE_PreInitStatus response.

The status values are defined as follows:

```
statusOk : 0
statusInvalidParameters : 1
statusPairingExpired : 2
statusSessionExpired : 3
```

12.4.3.4 pairingSlots

The pairingSlots attribute shall indicate the maximum number of pairing slots supported by a Sender.

12.4.3.5 sessionSlots

The sessionSlots attribute shall indicate the maximum number of session slots supported by a Sender.

12.4.3.6 vendorExtension

The vendorExtension attribute may contain vendor specific payload as described at section 12.5.

12.5 Vendor Extensions

The AKE_PreInit and AKE_PreInitStatus messages allocate 16 bytes for vendor specific extensions of the HKEP protocol. The content of the vendorExtension attribute of those messages is not specified by this technical recommendation. An optional behavior is proposed to allow a minimum interoperability across vendors.

A Sender should copy the value of the vendorExtension attribute from an AKE_PreInit message into the vendorExtension attribute of the AKE_PreInitStatus message response. This is the recommended cross-vendor behavior of a Sender.

A Sender could also respond with a value of the vendorExtension attribute in the AKE_PreInitStatus message which is different from the one received in an AKE_PreInit message. This is the recommended same-vendor behavior of a Sender.

Note: The vendorExtension space allocated within AKE_PreInit and AKE_PreInitStatus messages should allow vendors to easily add debugging extensions/information to their systems and help in reaching a high level of stability both in same-vendor and cross-vendor scenarios.

12.6 The receiver protocol

The following protocol shall be executed when the receiver attribute of an AKE_PreInit message is true.

The AKE_PreInit message shall always be the first message exchanged after a TCP/IP connection from a client to a server. The server shall be a Sender and the client shall be a Receiver. The AKE_PreInit.receiver attribute of the message shall be true.

The client and server instances of the TCP/IP connection shall enter the $HDCP_{RTP}$ v2.3 protocol right after the exchange of the AKE_PreInit and AKE_PreInitStatus messages (see sections 12.8 Figure 11, Figure 12 and 12.9 Figure 13, Figure 14 for associated server and client flowcharts).

The HDCP Transmitter interface port of a TCP/IP link shall assume a "Receiver Connected Indication" after the successful exchange of the AKE_PreInit and AKE_PreInitStatus messages, at the moment an AKE_Init message is sent by the Sender to the Receiver. At this time such Receiver may start subscribing to the associated HDCP Content from such Sender.

The HDCP Transmitter interface port of a TCP/IP link shall assume a "Receiver Disconnected Indication" when the HKEP session associated with an HDCP session expires.

An HKEP session shall bind a Receiver HDCP Receiver ID to a Sender NodeId and to an HDCP Content PortId. Those attributes are part of the AKE_PreInit message as receiverId, nodeId and portId respectively. Once an HKEP session becomes valid at a Sender (see section 13.2 for more details) it shall not expire unless it becomes inactive, or the associated HDCP session key is invalidated. Once an HKEP session becomes valid at a Receiver (see section 13.2 for more details) it shall not expire unless it becomes inactive, or the Receiver unsubscribes from the

associated HDCP Content. The validity of an HDCP session shall match the validity of an HKEP session.

An HKEP session shall become inactive at a Sender when the associated Receiver presents a Null Topology to the Sender, by sending an $HDCP_{RTP}$ v2.3 RepeaterAuth_Send_ReceiverID_List message with the DEVICE_COUNT and DEPTH attributes set to 0.

An HKEP session may become inactive at a Receiver at any time after the Receiver unsubscribes from the associated HDCP Content.

A Receiver shall close the TCP/IP connection when the HKEP or $HDCP_{RTP}$ v2.3 protocols fail, and it shall expire the associated HKEP session which remains invalid. A Receiver detects that the HKEP or $HDCP_{RTP}$ v2.3 protocols failed when the Sender closes the TCP/IP connection before an HDCP session becomes valid (see section 13.2 for more details).

A Sender shall close the TCP/IP connection when the HKEP or $HDCP_{RTP}$ v2.3 protocols fail, and it shall expire the associated HKEP session which remains invalid. A Sender detects that the HKEP or $HDCP_{RTP}$ v2.3 protocol failed when the Receiver closes the TCP/IP connection before an HDCP session becomes valid (see section 13.2 for more details).

A Sender may close the TCP/IP connection after a first successful exchange of the RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Stream_Manage messages with a Receiver once an HKEP session becomes valid (see section 13.2 for more details).

A Receiver may close the TCP/IP connection after a first successful exchange of the RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Stream_Manage messages with a Sender once an HKEP session becomes valid (see section 13.2 for more details).

12.6.1 With explicit pairing

When the AKE_PreInit.pairing flag is true, it indicates that the connection shall perform only the AKE stage of the $HDCP_{RTP}$ v2.3 protocol and terminate. When this flag is true, only the msg_id, version_*, pairing, receiver and vendorExtension attributes of the AKE_PreInit message are valid and the remaining attributes shall be ignored.

The Sender shall respond with an AKE_PreInitStatus message with the status attribute set to statusInvalidParameters if some of the AKE_PreInit attributes are invalid, and it shall close the TCP/IP connection.

Otherwise, the Sender shall respond with an AKE_PreInitStatus message with the status attribute set to statusPairingExpired, and it shall start the $HDCP_{RTP}$ v2.3 protocol by sending the AKE_Init message to the Receiver. If the Sender already has pairing information available for the Receiver, it should signal AKE_Stored_km to the Receiver. Otherwise it shall signal AKE_No_Stored_km to the Receiver.

The Receiver, after sending the AKE_PreInit message, shall wait for the AKE_PreInitStatus response, and it shall start the HDCP$_{RTP}$ v2.3 protocol by receiving the AKE_Init message. After sending the AKE_Send_Pairing_Info message, the Receiver should wait until the TCP/IP connection is closed by the Sender. Upon receiving the AKE_Send_Pairing_Info message, the Sender shall store the associated pairing information in a slot of an LRU cache in memory and close the TCP/IP connection. It should also store the associated pairing information in a secure non-volatile memory to recover it upon device restarts.

A Sender may support a limited number of pairing slots. The pairingSlots attribute of the AKE_PreInitStatus message shall indicate the maximum number of pairing slots available on the Sender. The Sender shall maintain an LRU cache of the pairing information using the AKE_PreInit.receiverId attribute as the key.

The sessionSlots attribute of the AKE_PreInitStatus message shall indicate the maximum number of session slots available on the Sender.

The pairing of Receivers with Senders should be performed "offline" such that the Senders and Receivers can accelerate the execution of the HDCP$_{RTP}$ v2.3 AKE protocol phase.

12.6.2 With implicit pairing

When the AKE_PreInit.pairing flag is false, it indicates that the connection shall perform the full the HDCP$_{RTP}$ v2.3 protocol. When this flag is false, all the attributes of the AKE_PreInit message are valid.

The Sender shall respond with an AKE_PreInitStatus message with the status attribute set to statusInvalidParameters if some of the AKE_PreInit attributes are invalid, and it shall close the TCP/IP connection.

Otherwise, the Sender shall respond with an AKE_PreInitStatus message with the status attribute set to statusPairingExpired if the pairing with the Receiver identified by the AKE_PreInit.receiverId attribute has not been performed or has expired. Then it shall start the HDCP$_{RTP}$ v2.3 protocol by sending the AKE_Init message to the Receiver.

Otherwise, the Sender shall respond with an AKE_PreInitStatus message with the status attribute set to statusSessionExpired if the Receiver identified by the AKE_PreInit.receiverId attribute attempts to join an expired/invalid HKEP session or the Receiver has set the restart/REAUTH_REQ flag. Then it shall start the HDCP$_{RTP}$ v2.3 protocol by sending the AKE_Init message to the Receiver. If the Sender has pairing information available for the Receiver it should signal AKE_Stored_km to the Receiver otherwise it shall signal AKE_No_Stored_km to the Receiver.

Otherwise, the Sender shall respond with an AKE_PreInitStatus message with the status attribute set to statusOk if the HKEP session is valid and has not expired. Then it shall restart an existing $HDCP_{RTP}$ v2.3 session at the Authentication with Repeaters stage.

A Receiver may connect to a Sender with the restart/REAUTH_REQ flag of the AKE_PreInit message set to true at any time. It could be because it restarted, it reached the $HDCP_{RTP}$ v2.3 unauthenticated state, or because of any other reason.

When a Receiver sets the AKE_PreInit restart/REAUTH_REQ flag to true, or the AKE_PreInitStatus response has the status attribute set to statusPairingExpired or statusSessionExpired, the topology information stored in an active HKEP session of the Sender shall remain effective, even if the HDCP protocol restarts at the AKE_Init stage. Only a fully authenticated HDCP Repeater shall be allowed to change the state (topology) of an active $HDCP_{RTP}$ v2.3 session.

The Receiver, after sending the AKE_PreInit message, shall wait for the AKE_PreInitStatus response to detect a success or failure as follow.

If the AKE_PreInitStatus response has the status attribute set to statusInvalidParameters, the Receiver shall close the TCP/IP connection.

Otherwise if the AKE_PreInitStatus response has the status attribute set to statusPairingExpired or statusSessionExpired, the Receiver shall start the $HDCP_{RTP}$ v2.3 protocol by receiving the AKE_Init message. After receiving the AKE_Send_Pairing_Info message, the Sender shall store the associated pairing information in a slot of an LRU cache in memory. It should also store the associated pairing information in a secure non-volatile memory to recover it upon device restarts. A Sender may support a limited number of pairing slots. The pairingSlots attribute of the AKE_PreInitStatus message shall indicate the maximum number of pairing slots available on the Sender. The Sender shall maintain an LRU cache of the pairing information using the AKE_PreInit.receiverId attribute as the key.

Otherwise if the AKE_PreInitStatus response has the status attribute set to statusOk, the Receiver shall restart an existing $HDCP_{RTP}$ v2.3 session at the Authentication with Repeaters stage.

A Receiver subscribing to HDCP Content produced by a Sender shall connect to the Sender and shall send an AKE_PreInit message filled with the receiverId attribute set to the Receiver HDCP Receiver ID, the nodeId attribute set to the NodeId of the Sender and the portId attribute set to the PortId of the subscribed HDCP Content. The Sender shall respond with an AKE_PreInitStatus message indicating that the receiver is joining a new HKEP session (statusPairingExpired or statusSessionExpired), or an active HKEP session (statusOk).

The nodeId attribute of the AKE_PreInit message shall identify the Sender targeted by the connection and producing the HDCP Content. It shall uniquely identify the Sender and correspond to the node-id parameter of the SDP transport file "hkep" session attribute.

The portId attribute of the AKE_PreInit message shall identify the HDCP Content targeted by the connection and consumed by the Receiver, and shall correspond to the port-id parameter of the SDP transport file "hkep" session attribute.

The receiverId attribute of the AKE_PreInit message shall correspond to the Receiver HDCP Receiver ID establishing the connection and consuming the HDCP Content.

Senders and Receivers shall cache their active HKEP sessions. A Sender shall reuse a cached session unless it becomes inactive or expired. A Receiver shall reuse a cached session unless the session becomes inactive, expired, the Receiver sets the restart/ REAUTH_REQ flag on an AKE_PreInit message, or the status attribute of an AKE_PreInitStatus response from a Sender is not set to statusOk.

Senders and Receivers may support a limited number of session slots. The attribute sessionSlots of the AKE_PreInitStatus message shall indicate the maximum number of slots available on the Sender. The Sender shall maintain an LRU cache of the session information using the AKE_PreInit.receiverId and AKE_PreInit.portId attributes as the key. The Receiver shall maintain a cache of the session information using the AKE_PreInit.nodeId and AKE_PreInit.portId attributes as the key.

A Receiver should make its HKEP session inactive at a Sender when it no longer subscribed to the associated HDCP Content by sending an $HDCP_{RTP}$ v2.3 RepeaterAuth_Send_ReceiverID_List message with the DEVICE_COUNT and DEPTH attributes set to 0 (Null Topology). Only a fully authenticated HDCP Repeater shall be allowed to change the state (topology) of an active $HDCP_{RTP}$ v2.3 session.

Note: A Receiver that unsubscribes from the HDCP Content of a Sender but that does not terminate its HKEP sessions, consumes resources on the Sender and remains in the downstream topology observed by the Sender. This may become problematic if multiple Receivers do not make their HKEP session inactive on the Sender.

12.6.3 With reconnect

A Receiver should reconnect to a Sender with the restart/REAUTH_REQ attribute of the AKE_PreInit message set to false after an HKEP session has become valid (see section 13.2 for more details) when topology information associated with an HKEP session has changed, the streamCtr value received along with an encrypted HDCP Content stream associated with an HKEP session is not found in the last RepeaterAuth_Stream_Manage message received from the Sender, or to make an HKEP session inactive.

When a Receiver reconnects to a Sender with the restart/REAUTH_REQ attribute of the AKE_PreInit message set to false, it shall either send an $HDCP_{RTP}$ v2.3 RepeaterAuth_Send_ReceiverID_List message (new topology information) or an $HDCP_{RTP}$ v2.3 Null message (no new topology information). The Sender reconnecting to an active HKEP session with a Receiver shall either send an $HDCP_{RTP}$ v2.3 RepeaterAuth_Stream_Manage message (new stream types), or an $HDCP_{RTP}$ v2.3 Null message (no new stream types).

The Sender and Receiver may close the TCP/IP connection after successfully exchanging RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Stream_Manage messages with each other.

Note: The objective of the $HDCP_{RTP}$ v2.3 Null message (msg_id 1) is to quickly indicate to the peer that there is no new information available (see section 13.2 for more details).

12.6.4 Protocol illustration with explicit pairing (informative)

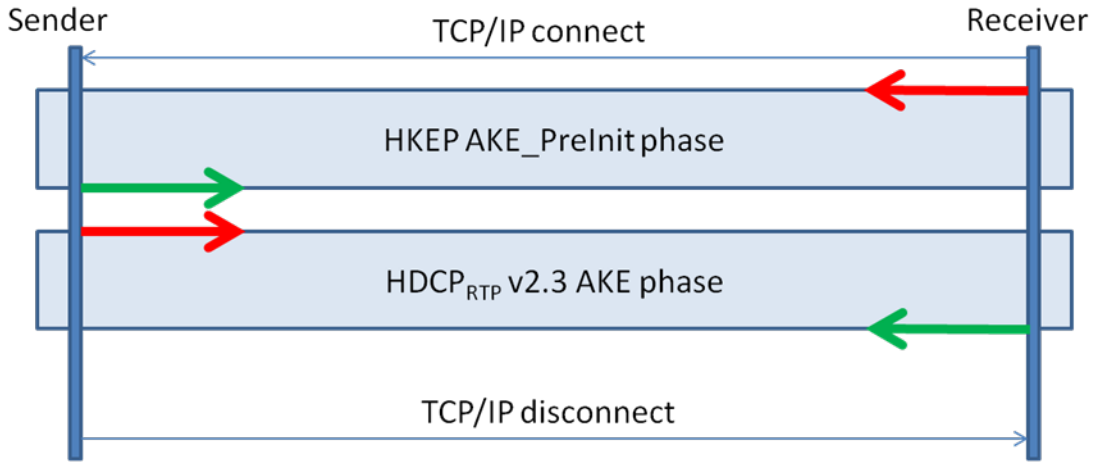

Figure 7 - Protocol with explicit pairing

12.6.5 Protocol illustration with implicit pairing (informative)

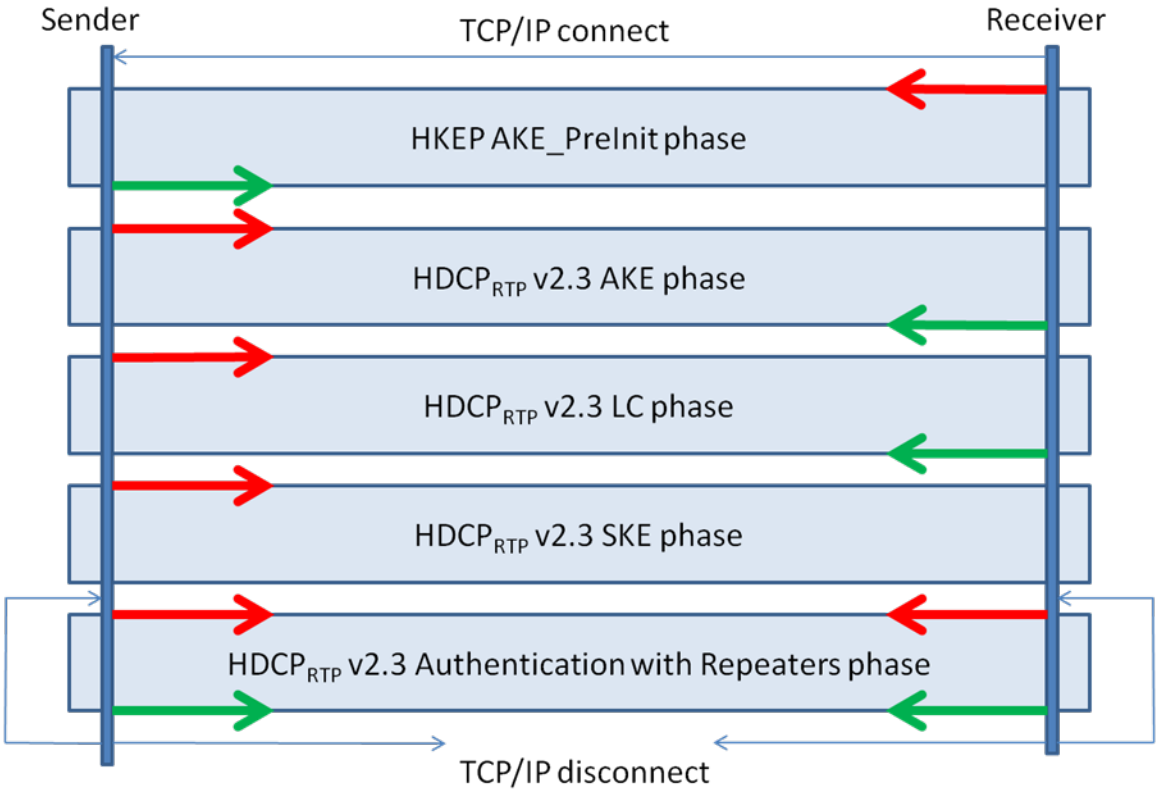

Figure 8 - Protocol with implicit pairing

12.6.6 Protocol illustration with reconnect (informative)

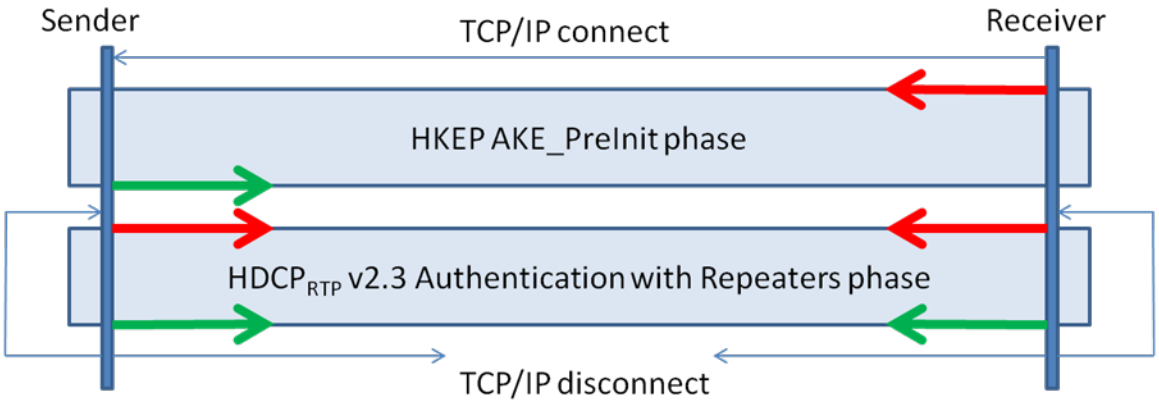

Figure 9 - Protocol with reconnect

12.7 The non-receiver protocol

The following protocol shall be executed when the receiver attribute of an AKE_PreInit message is false.

Note: In an NMOS environment, an AMWA NMOS controller may use this protocol to inspect the HKEP pairingSlots and sessionSlots capabilities of a Sender.

The pairingSlots and sessionSlots attributes of the AKE_PreInitStatus message shall indicate the maximum number of slots supported by the device.

Note: There is no mechanism for retrieving the actual number of slots available at any given time.

The AKE_PreInit message shall be the first message exchanged after a TCP/IP connection from a client to a server. The server shall be a Sender and the client could be of any type. The receiver attribute of the message shall be false and the pairing attribute shall be true.

The client and server instances of the TCP/IP connection shall only exchange the AKE_PreInit and AKE_PreInitStatus messages (see sections 12.8 and 12.9 for associated server and client flowcharts).

Only the msg_id, version_*, pairing, receiver and vendorExtension attributes of the AKE_PreInit message are valid and the remaining attributes shall be ignored.

If some of the AKE_PreInit attributes are invalid, the Sender shall respond with an AKE_PreInitStatus message with the status attribute set to statusInvalidParameters, and shall close the TCP/IP connection.

If all the AKE_PreInit attributes are valid, the Sender shall respond with an AKE_PreInitStatusmessage with status attribute set to statusOk, and the pairingSlots and sessionSlots attributes shall indicate the maximum device capabilities.

Note: The pairing flag indicates that the connection performs the AKE stage of the protocol and terminate, but as the connecting device is not a Receiver, there is no such AKE stage to execute, and so the Sender will close the TCP/IP connection after sending the AKE_PreInitStatus response.

12.7.1 Protocol illustration for non-receivers (informative)

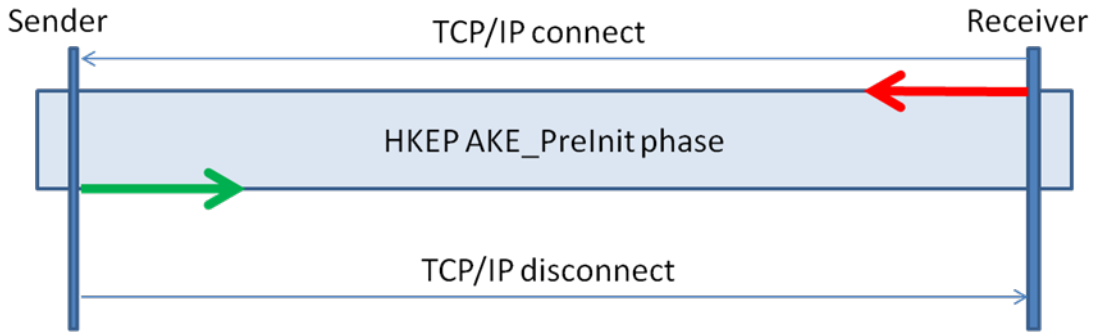

Figure 10 - Protocol for non-receivers

12.8 Sender (server side) protocol flowcharts (informative)

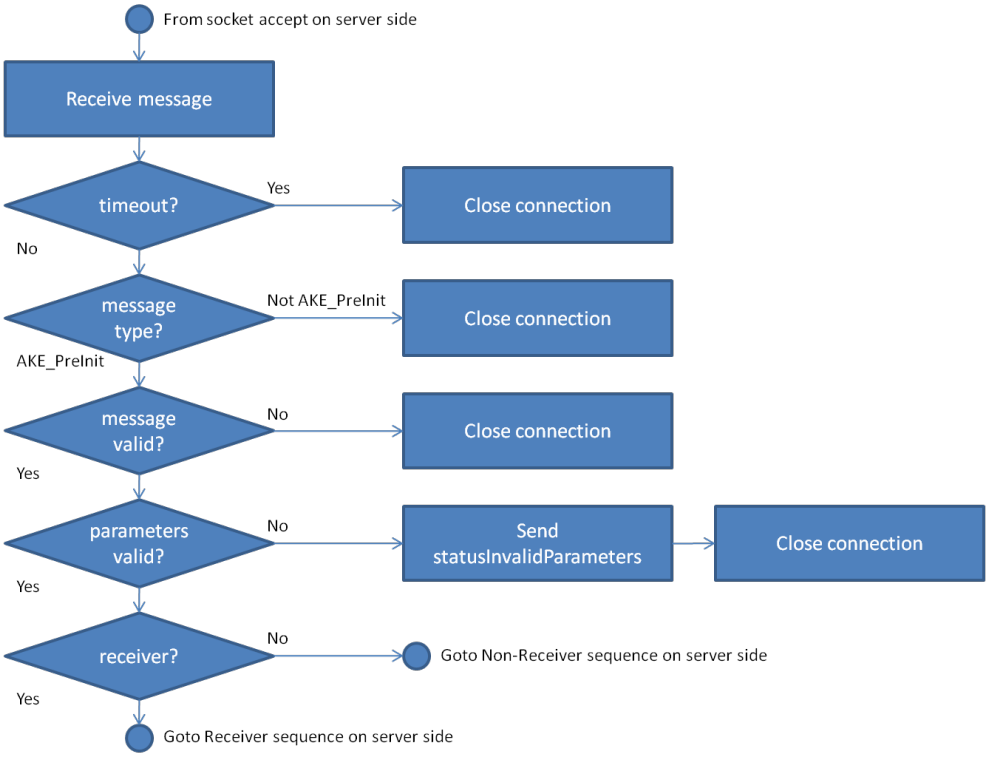

Figure 11 - Sender (server side) protocol

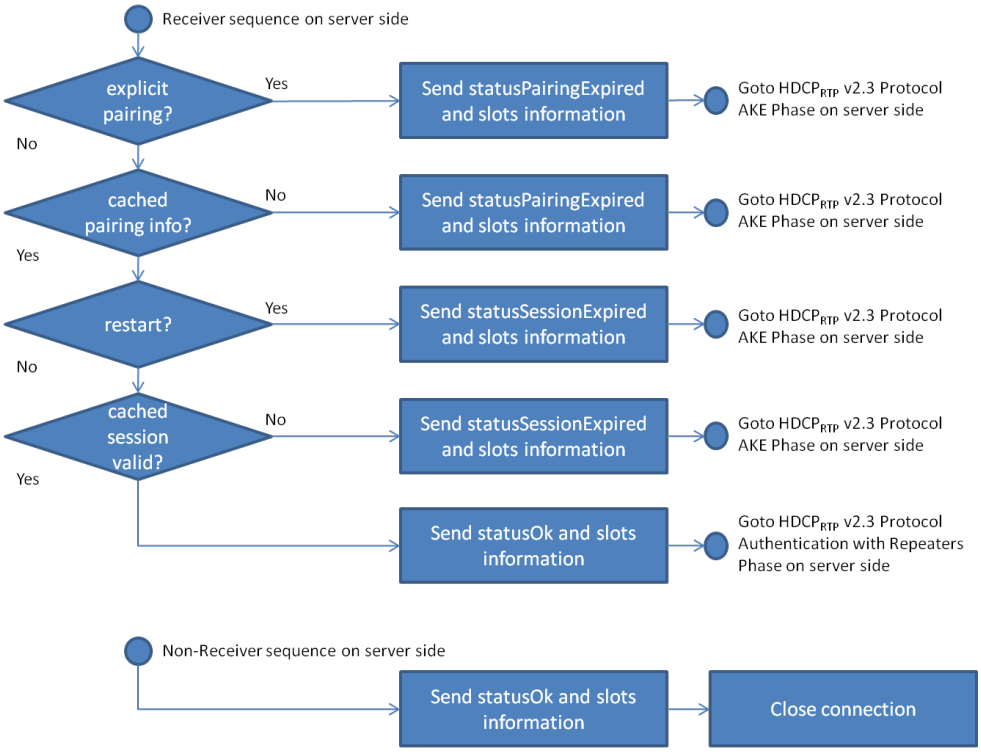

Figure 12 - Sender (server side) protocol

12.9 Receiver (client side) protocol flowcharts (informative)

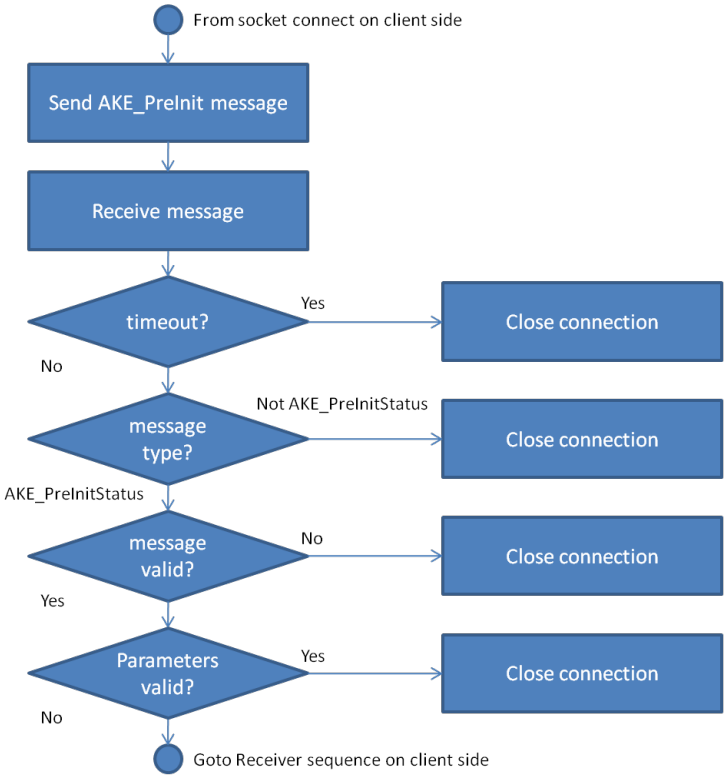

Figure 13 - Receiver (client side) protocol

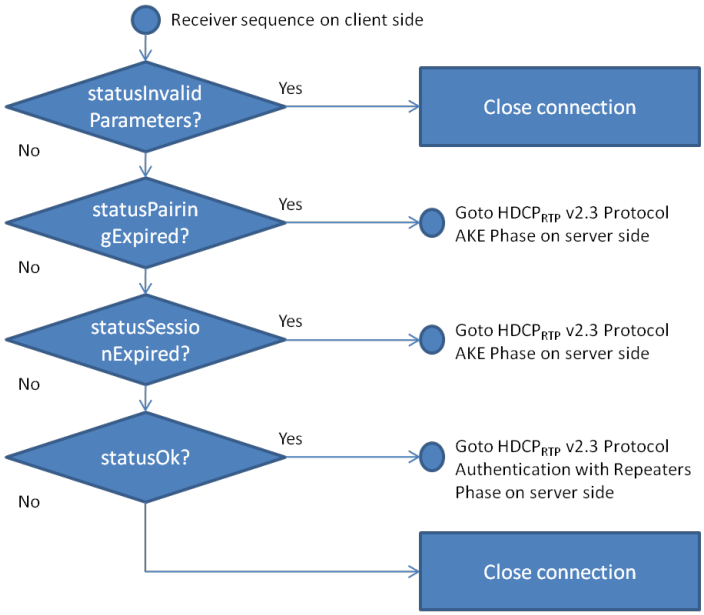

Figure 14 - Receiver (client side) protocol

12.10 Protocol sequence of a TCP/IP Connection (informative)

A TX_{ip} Port on a Sender listens to incoming Receiver connections. Each connection starts a new instance of the HKEP protocol. Once the initial AKE_PreInit message has been processed and properly acknowledged with an AKE_PreInitStatus message, the HKEP protocol transports the messages of the $HDCP_{RTP}$ v2.3 protocol.

An HDCP Session as specified in the $HDCP_{RTP}$ v2.3 specification takes place in a TCP/IP connection when the $HDCP_{RTP}$ v2.3 protocol executes successfully and ends when the associated HKEP session expires.

In the following table some symbols are used to illustrate the flow of information: → sending a message/response from the Sender to the Receiver, ← sending a message/response from the Receiver to the Sender, sW sender waiting for a connection, a message or a response, rW receiver, waiting for a message or a response.

Step	**Sender**	**→ ←**	**Receiver**
0	listen		
1	accept	sW	
2	Spawn a new HKEP process at state 4	←	connect
3	goto 1		
4	Receives AKE_PreInit from socket The operation times out after ProtocolTimeout seconds and closes the connection. If the message type is not AKE_PreInit close the connection. Verify the message. On error close the connection. Check the AKE_PreInit message parameters. On error return the AKE_PreInitStatus message with statusInvalidParameters and close the connection. If the pairing attribute is true and the receiver attribute is true, return the AKE_PreInitStatus message at step 5 with statusPairingExpired and start the $HDCP_{RTP}$ v2.3 authentication protocol by sending the AKE_Init message at step 55. If the pairing attribute is true and the receiver attribute is false, return the AKE_PreInitStatus message at step 5	sW	

	with statusOk and close the connection at step 60. If the pairing attribute is false, check for pairing information for the incoming receiverId. If not found, return the AKE_PreInitStatus message at step 5 with statusPairingExpired and start the HDCP$_{RTP}$ v2.3 authentication protocol by sending the AKE_Init message at step 55. Otherwise check for a cached HKEP session. If it is expired/invalid return the AKE_PreInitStatus message at step 5 with statusSessionExpired and start the HDCP$_{RTP}$ v2.3 authentication protocol by sending the AKE_Init message at step 55. Otherwise return the AKE_PreInitStatus message at step 5 with statusOk and restart a previous session at step 58.		
		←	Construct an AKE_PreInit message and send it to the sender.
5	Send the AKE_PreInitStatus message. If the status is statusPairingExpired or statusSessionExpired start a fresh protocol with AKE at step 55 if the receiver attribute is true, otherwise if the status is statusOk restart a previous session at step 58, otherwise close the connection.	→	
		rW	Receive AKE_PreInitStatus The operation times out after ProtocolTimeout seconds and closes the connection. If the message type is not AKE_PreInitStatus close the connection. If the status is statusSessionExpired or statusPairingExpired start a fresh protocol with AKE at step 55, otherwise if the status is statusOk restart a previous session at step 58, otherwise close the connection.
55			
AKE as per HDCP$_{RTP}$ v2.3 specification			
56			
LC as per HDCP$_{RTP}$ v2.3 specification			
57			
SKE as per HDCP$_{RTP}$ v2.3 specification			
58			
Authentication with Repeaters as per HDCP$_{RTP}$ v2.3 specification			

59	Close the connection and be done.		
60	branch here on error Expires the HKEP session if not already valid. Close the connection		branch here on error Expires the HKEP session if not already valid. Close the connection

13 The HDCP$_{RTP}$ v2.3 protocol

This section describes how the optional behaviors / aspects of the HDCP$_{RTP}$ v2.3 specification shall be implemented by HKEP compliant devices.

13.1 Locality check

The locality check sequence described at section 2.3 of the HDCP$_{RTP}$ v2.3 specification shall be performed as follows by HKEP compliant devices.

Senders and Receivers shall set the flags TRANSMITTER_LOCALITY_PRECOMPUTE_SUPPORT and RECEIVER_LOCALITY_PRECOMPUTE_SUPPORT to true in the AKE_Transmitter_Info and AKE_Receiver_Info messages.

In the case of a locality check failure due to a mismatch of L and L' at the HDCP Transmitter, the locality check shall be reattempted by the HDCP Transmitter for a maximum of 1023 additional attempts (for a maximum allowed 1024 total trials) with the transmission of an LC_Init message containing a new r_n.

The Sender shall send a new LC_Init message as soon as it observes a mismatch of L and L' from the LC_Send_L_prime message. If the Sender does not receive the LC_Send_L_prime message within ProtocolTimeout milliseconds of the sending of the RTT_Challenge message, it shall abort the protocol and shall close the TCP/IP connection.

13.2 Authentication with repeaters

The authentication with repeaters sequence described in section 2.5 of the HDCP$_{RTP}$ v2.3 specification shall be performed as follows by HKEP compliant devices.

Refer to the HDCP$_{RTP}$ v2.3 specification for the applicable state diagrams:

- Figure 2.16. HDCP Repeater Downstream Authentication Protocol State Diagram
- Figure 2.17. HDCP Repeater Upstream Authentication Protocol State Diagram
- Figure 2.13. HDCP Transmitter Authentication Protocol State Diagram

Authentication with repeaters and stream content management shall be implemented in parallel with the flow of encrypted HDCP Content and HDCP link synchronization.

The content stream management shall be performed as follows:

On the Sender the F9 state shall be implemented on the transitions from F6 to F7 {F6-F9-F7}, from F5 to F7 (if receiving RepeaterAuth_Send_ReceiverID_List) {F5-F9-F7} or from F5 to F5 (if receiving an HDCP$_{RTP}$ v2.3 Null message) {F5-F9-F5}.

On the Receiver the state C7 shall be implemented on the transition from C5 to C6 (if sending RepeaterAuth_Send_ReceiverID_List) {C5-C7-C6} or from C8 to C8 (if sending an HDCP$_{RTP}$ v2.3 Null message) {C8-C7-C8}.

For pure HDCP Transmitter the A9 state shall be implemented on the transition from A6 to A7 {A6-A9-A7}, from A5 to A7 (if receiving RepeaterAuth_Send_ReceiverID_List) {A5-A9-A7} or from A5 to A5 (if receiving an HDCP$_{RTP}$ v2.3 Null message) {A5-A9-A5}.

Note: The objective for specifying the precise sequence involving the content stream management is to ensure that an HKEP session becomes valid on a Receiver only if it is also valid on the Sender. When receiving the RepeaterAuth_Send_Ack message a Receiver knows for sure that the HKEP session on the Sender became valid because the Sender does not return this message before it successfully receives a RepeaterAuth_Stream_Ready message from the Receiver.

13.2.1 Exchange sequence

Refer to Figure 15 of section 13.2.5 for an illustration of the exchange sequence described in the following paragraphs.

A Receiver executing the Authentication with Repeaters phase of the HDCP$_{RTP}$ v2.3 protocol shall initially either send an HDCP$_{RTP}$ v2.3 RepeaterAuth_Send_ReceiverID_List message or an HDCP$_{RTP}$ v2.3 Null message. A Sender shall initially either send an HDCP$_{RTP}$ v2.3 RepeaterAuth_Stream_Manage message or an HDCP$_{RTP}$ v2.3 Null message. After sending their initial message, the Receiver and the Sender shall attempt to receive the initial message from their peer. If the receive operation does not complete within ProtocolTimeout milliseconds, the TCP/IP connection shall be closed. Otherwise, the Receiver and Sender shall process the message received.

A Receiver executing the Authentication with Repeaters phase of the HDCP$_{RTP}$ v2.3 protocol shall send a RepeaterAuth_Stream_Ready message if it initially received a RepeaterAuth_Stream_Manage message. Then it shall attempt to receive the RepeaterAuth_Send_Ack message, if it initially sent a RepeaterAuth_Send_ReceiverID_List message.

A Sender executing the Authentication with Repeaters phase of the HDCP$_{RTP}$ v2.3 protocol shall attempt to receive a RepeaterAuth_Stream_Ready message if it sent an initial RepeaterAuth_Stream_Managemessage. Then it shall send a RepeaterAuth_Send_Ack message if it initially received a RepeaterAuth_Send_ReceiverID_List message.

A Receiver should send a Receiver_AuthStatus message as required by the $HDCP_{RTP}$ v2.3 specification but a Sender shall not use it to establish the validity if an HKEP or $HDCP_{RTP}$ v2.3 session. A Sender shall ignore the message if it receives it, as allowed by the $HDCP_{RTP}$ v2.3 specification. A Receiver communicates the $HDCP_{RTP}$ v2.3 REAUTH_REQ state to the Sender through the AKE_PreInit message using the restart/REAUTH_REQ flag. A Receiver shall close the TCP/IP connection either instead of, or after sending a Receiver_AuthStatus message with REAUTH_REQ set to true. A Receiver shall not send a Receiver_AuthStatus message with REAUTH_REQ set false.

13.2.2 A first successful exchange

A Sender shall mark the current HKEP session as valid after a first successful exchange of the topology (RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Send_Ack messages) and stream management information (RepeaterAuth_Stream_Manage and RepeaterAuth_Stream_Ready messages) with the Receiver. This marking shall happen after successfully sending the RepeaterAuth_Send_Ack message to the Receiver while executing the sequence described at section 13.2.1. Once an HKEP session becomes valid, and therefore, an $HDCP_{RTP}$ v2.3 session has become valid, timeouts, disconnects or protocol errors shall not invalidate the session. A Sender may invalidate an inactive HKEP session at any time, but it shall not invalidate active HKEP sessions unless it also invalidates the associated $HDCP_{RTP}$ v2.3 session key.

A Receiver shall mark the current HKEP session as valid after a first successful exchange of the topology (RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Send_Ack messages) and stream management information (RepeaterAuth_Stream_Manage and RepeaterAuth_Stream_Ready messages) with the Sender. This marking shall happen after receiving successfully the RepeaterAuth_Send_Ack message from the Sender while executing the sequence described at section 13.2.1. Once an HKEP session becomes valid, and therefore, an $HDCP_{RTP}$ v2.3 session has become valid, timeouts, disconnects or protocol errors shall not invalidate the session.

A Receiver shall not decrypt an encrypted HDCP Content stream until its HKEP session becomes valid. The SKE stage completes before the HKEP session becomes valid, but it is only after a first successful exchange of the RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Stream_Manage messages, that the HKEP session becomes valid. Only then shall decryption of an encrypted HDCP Content stream start.

13.2.3 A subsequent successful exchange

After a first successful exchange of the topology and stream management information with a Receiver, a Sender shall consider a subsequent exchange successful when one of the following conditions occurs:

- When it has successfully received the Null message from the Receiver and it has successfully sent the Null message to the Receiver,
- When it has successfully received the Null message from the Receiver and it has successfully performed an exchange of the RepeaterAuth_Stream_Manage and RepeaterAuth_Stream_Ready messages with the Receiver,
- When it has performed a successful exchange of the RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Send_Ack messages with the Receiver and it has successfully sent the Null message to the Receiver,
- When it has performed a successful exchange of the RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Send_Ack messages with the Receiver and it has successfully performed an exchange of the RepeaterAuth_Stream_Manage and RepeaterAuth_Stream_Ready messages with the Receiver.

The exact moment when such subsequent exchange is considered successful shall be after successfully executing the sequence described in section 13.2.1.

After a first successful exchange of the topology and stream management information with a Sender, a Receiver shall consider a subsequent exchange successful when one of the following conditions occurs:

- When it has successfully received the Null message from the Sender and it has successfully sent the Null message to the Sender,
- When it has successfully received the Null message from the Sender and it has performed a successful exchange of the RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Send_Ack messages with the Sender,
- When it has performed a successful exchange of the RepeaterAuth_Stream_Manage and RepeaterAuth_Stream_Ready messages with the Sender and it has successfully sent the Null message to the Sender,
- When it has performed a successful exchange of the RepeaterAuth_Stream_Manage and RepeaterAuth_Stream_Ready messages with the Sender and it has performed a successful exchange of the RepeaterAuth_Send_ReceiverID_List and RepeaterAuth_Send_Ack messages with the Sender.

The exact moment when such subsequent exchange is considered successful shall be after successfully executing the sequence described in section 13.2.1.

13.2.4 Null Topology

A Receiver unsubscribing from the HDCP Content of a Sender may choose to make its HKEP session inactive at the Sender by sending an RepeaterAuth_Send_ReceiverID_List message with the DEVICE_COUNT and DEPTH attributes set to 0 (Null Topology) during the execution of the sequence described in section 13.2.1. As a result the Receiver will no longer self-subscribe to

the HDCP Content, it will no longer be part of the topology tree associated with the HDCP Content, and it will perform a transition to state C0 in the "Figure 2.17. HDCP Repeater Upstream Authentication Protocol State Diagram" of the HDCP$_{RTP}$ v2.3 protocol.

13.2.5 Exchange sequence illustration (informative)

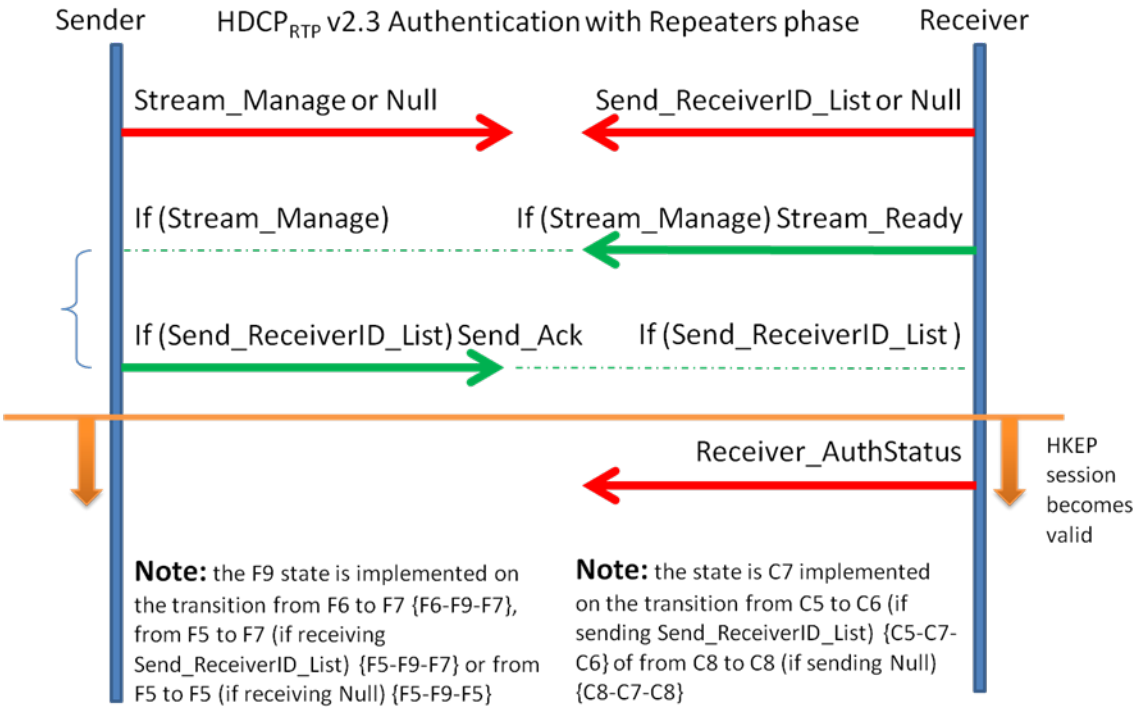

Figure 15 - Exchange sequence

13.3 RepeaterAuth_Stream_Manage

The RepeaterAuth_Stream_Manage message described in section 4.3.15 of the HDCP$_{RTP}$ v2.3 specification shall be constructed as follows by HKEP compliant devices.

A Sender shall construct a RepeaterAuth_Stream_Manage message as follow:

- For each content stream of a given session, associate a unique streamCtr value to a Type value and to one or multiple ContentStreamID values.
- The Type and ContentStreamID values associated with a streamCtr value shall be immutable for the duration of a session. A given streamCtr value shall be associated withgiven Type and ContentStreamID values once for the lifetime of the Sender session key.

Note: For example, if streamCtr value 1 is associated with { Type: 0, ContentStreamID: 0xabcdef } it cannot later in time become associated with { Type: 1, ContentStreamID: 0xabcdef }. A new streamCtr value must be used for the new association. Receivers should be aware that a

streamCtr value may be associated with multiple ContentStreamID values. This scenario may arise when the definition of ContentStreamID relates to some transport channel identifier and a given content stream is transmitted over multiple transport channels.

The attribute k of the RepeaterAuth_Stream_Manage message shall be bounded to a maximum value of (2 * ProtocolMaxContentStreams). The objective is to have a practical maximum size for the RepeaterAuth_Stream_Manage message. There shall not be more than ProtocolMaxContentStreams different values of streamCtr in a given RepeaterAuth_Stream_Manage message, which effectively limits the number of audio/video streams to half the range allowed for k. The remaining entries allow the association of a streamCtr value to multiple ContentStreamID values.

14 $HDCP_{RTP}$ v2.3 media stream private data

A Sender shall embed the following information in each ST 2110/IPMX media stream through RTP header extensions as specified in the $HDCP_{RTP}$ v2.3 specification. The HDCP Full IV Counters extension header provides all the frz flag, streamCtr and inputCtr values. The HDCP Short IV Counters extension header provides only the least significant bit of the inputCtr value.

State	Description
frz	Indicate that the byte stream is not encrypted by k_s (cipher disabled)
streamCtr	Uniquely identify the byte stream.
inputCtr	Position within the byte stream of the start of the current picture, buffer, sequence of bytes, etc...

The HDCP Full and Short IV Counters extension headers shall be the first header extension of an RTP packet.

When the frz flag is false, it shall indicate that the HDCP Content is encrypted with the HDCP session key. When frz is true, it shall indicate that the content is unencrypted.

The streamCtr and inputCtr values shall be used by the stream cipher to encrypt/decrypt the HDCP Content. The streamCtr shall also be used to associate stream management information to a ST 2110/IPMX media stream and to signal Receivers about Sender state changes.

14.1 Signaling the receivers

A Receiver shall monitor the streamCtr value received along with a subscribed ST 2110/IPMX media stream. A change of streamCtr value indicates that the state of the associated Sender has changed. If a Receiver is not in possession of the stream management information related to the new streamCtr value, it shall connect or reconnect with the Sender and execute the $HDCP_{RTP}$ v2.3 protocol.

A Sender shall change the streamCtr value of a content stream to signal that the associated stream management information has changed.

A Sender should change the streamCtr values of all the content streams of an $HDCP_{RTP}$ v2.3 session to signal that it has transitioned to the H1/P1 state from either the A7/F7 or A9/F9 states. Refer to $HDCP_{RTP}$ v2.3 specification figures 2.12, 2.13, 2.15, 2.16 for the applicable state diagrams.

A Sender should change the streamCtr values of all the content streams of an $HDCP_{RTP}$ v2.3 session to signal that it has changed or invalidated the session key it uses to encrypt HDCP Content.

15 Encrypted HDCP Content (informative)

The $HDCP_{RTP}$ v2.3 specification states that the RTP Header, RTP Header Extensions and RTP Payload Header are not encrypted. As per RFC 8088 (How to Write an RTP Payload Format) The RTP Payload Header is defined as follows: "RTP payload formats often need to include metadata relating to the payload data being transported. Such metadata is sent as a payload header, at the start of the payload section of the RTP packet. The RTP packet also includes space for a header extension (see RFC5285); this can be used to transport payload format independent metadata, for example, an SMPTE time code for the packet (see RFC5484). The RTP header extensions are not intended to carry headers that relate to a particular payload format, and must not contain information needed in order to decode the payload."

Depending on the RTP Payload Format and as specified in the associated RFC specification, the first N bytes of the RTP Payload may contain an RTP Payload Header, as defined in the RFC payload format specification, and as such is not encrypted. If the RFC payload format specification does not define an RTP Payload Header, the full RTP Payload is encrypted.

The mechanism used by a Sender and a Receiver to detect which portion of the RTP Payload is encrypted is outside the scope of this specification. It is expected that an AMWA NMOS message and/or an SDP transport file associated with a Sender media stream would provide the necessary payload format information to a Receiver.

Note: The $HDCP_{RTP}$ v2.3 specification has provisions for the encryption of audio and video content only. It follows that ST 2110/IPMX ancillary data streams are not HDCP encrypted.