High-bandwidth Digital Content Protection System

Mapping HDCP to WHDI

Revision 2.0

21 October, 2008

Digital Content Protection LLC

Notice

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

Web: www.digital-cp.com

Revision History

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 2.00.

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 2.0-compliant Devices must interoperate with other HDCP 2.0-compliant Devices 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 technology license. Additionally, HDCP Transmitters may be subject to additional robustness and compliance rules associated with other content protection technologies.

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.

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 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 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*.

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.

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.

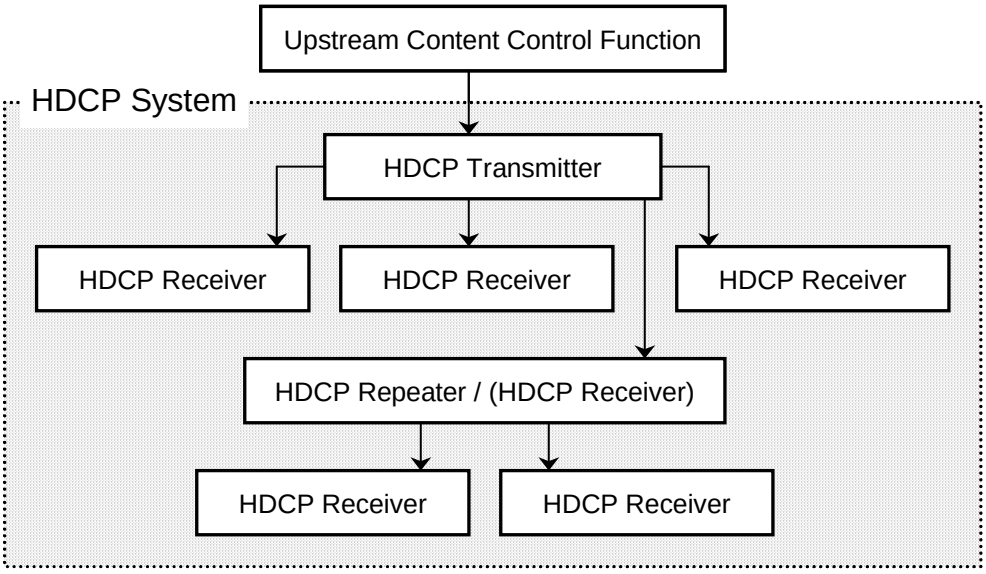


Figure 1.1. Sample Connection Topology of an HDCP System

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 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 $kpub_{dcp}$.

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
Reserved	16	3087:3072	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 $kpriv_{rx}$. 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.

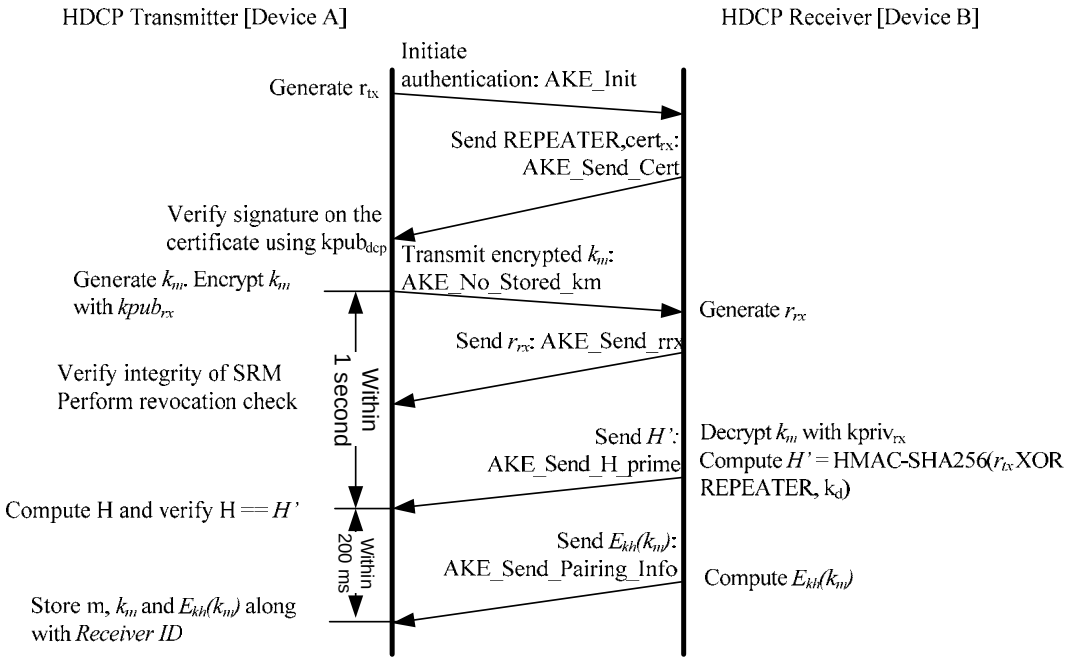


Figure 2.1. Authentication and Key Exchange (Without Stored k_m)

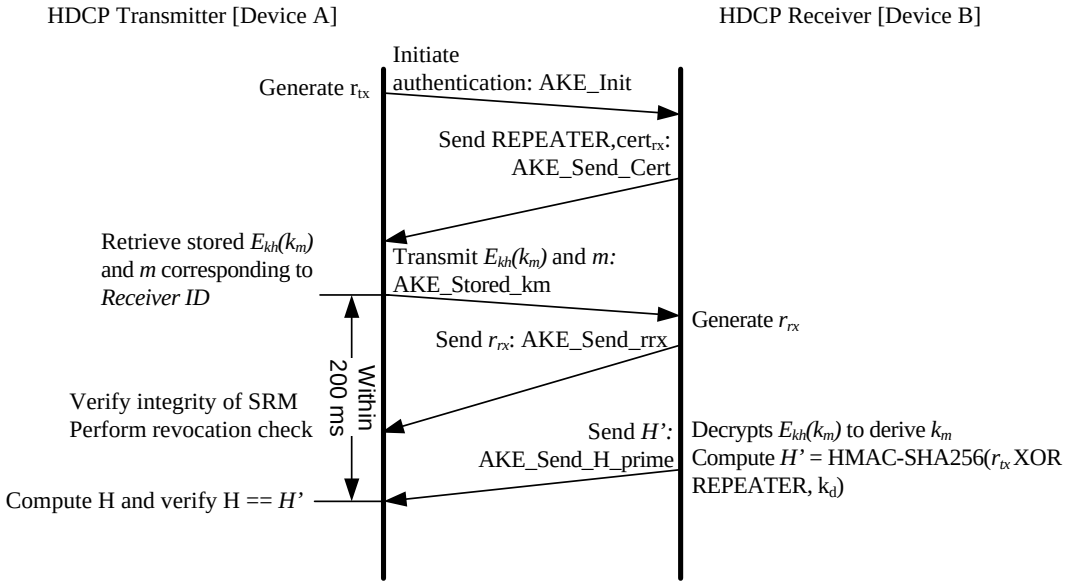


Figure 2.2. Authentication and Key Exchange (With Stored k_m)

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

- 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 $kpub_{dcp}$. Failure of signature verification constitutes an authentication failure and the HDCP Transmitter aborts the authentication protocol (See Section 2.7 on handling authentication failures).

        - Generates a pseudo-random 128-bit master key k_m. Encrypts k_m with $kpub_{rx}$ ($E_{kpub}(km)$) and sends AKE_No_Stored_km message to the receiver containing the 1024-bit $E_{kpub}(km)$. 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 $kpub_{dcp}$. Failure of this integrity check constitutes an authentication failure and causes the HDCP Transmitter to abort authentication protocol (See Section 2.7 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 2.7 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 2.8 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 = HMAC-SHA256(r_{tx} XOR REPEATER, k_d) where HMAC-SHA256 is computed over r_{tx} XOR REPEATER and the key used for HMAC is k_d. REPEATER is XORed with the least significant byte of r_{tx}.

        - Receives AKE_Send_H_prime message from the receiver containing the 256-bit *H'*. This message must be received within one second after sending $E_{kpub}(km)$ (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 2.7 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 $kpub_{dcp}$. 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 2.8 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 = HMAC-SHA256(r_{tx} XOR REPEATER, k_d) where HMAC-SHA256 is computed over r_{tx} XOR REPEATER and the key used for HMAC is k_d. REPEATER is XORed with the least significant byte of r_{tx}.
  - 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 2.7 on handling authentication failures).

The HDCP 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 HDCP Receiver
    - Decrypts k_m with $kpriv_{rx}$ using RSAES-OAEP decryption scheme.
    - Performs key derivation as explained in Section 2.8 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 *H'* = HMAC-SHA256(r_{tx} XOR REPEATER, k_d). 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.

o If AKE_Stored_km is received, the HDCP Receiver

- Computes 128-bit k_h = SHA-256($kpriv_{rx}$)[127:0]
- Decrypts $E_{kh}(k_m)$ 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.
- Performs key derivation as explained in Section 2.8 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 *H'* = HMAC-SHA256(r_{tx} XOR REPEATER , k_d). 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 $kpriv_{rx}$, 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 = SHA-256($kpriv_{rx}$)[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_{tx} 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 2.7 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 ID*s of all HDCP Receivers with which pairing was implemented by the HDCP Transmitter.

Figure 2.3 illustrates the encryption of k_m with k_h.

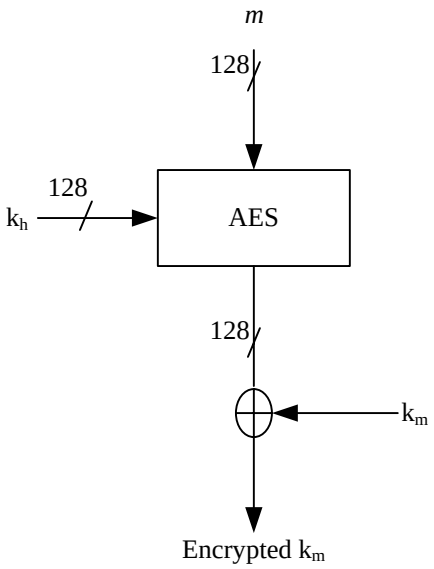


Figure 2.3. $E_{kh}(k_m)$ Computation

128-bit *m* is constructed by appending 64 0s to r_{tx}. r_{tx} 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. 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 = HMAC-SHA256(r_n , k_d XOR r_{rx}) where HMAC-SHA256 is computed over r_n and the key used for HMAC is k_d 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'*.

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 illustrates locality check between the HDCP Transmitter and HDCP Receiver.

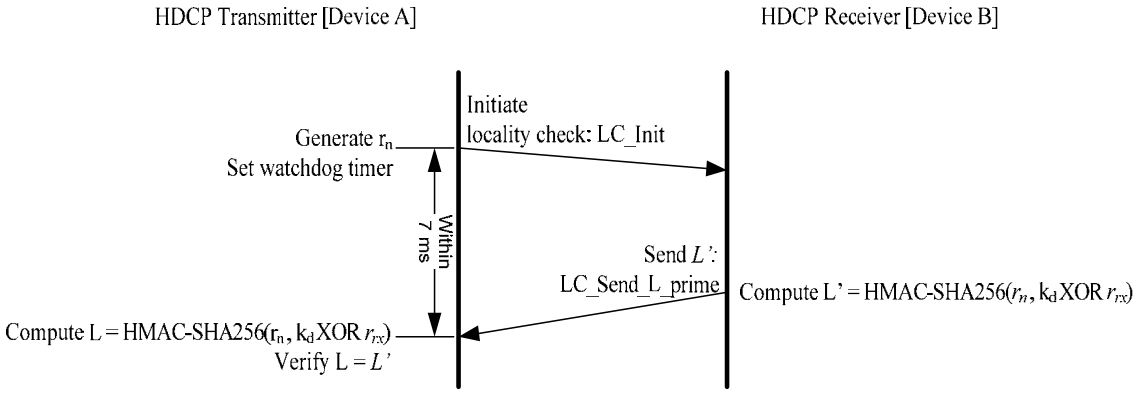


Figure 2.4. Locality Check between HDCP Transmitter and HDCP Receiver

An HDCP Receiver

- Computes a 256-bit value L' = HMAC-SHA256(r_n, k_d XOR r_{rx}).
- Sends LC_Send_L_prime message containing 256-bit 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 2.7 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}.
- Performs key derivation as explained in Section 2.8 to generate 128-bit $dkey_2$ where $dkey_2$ is the derived key when ctr =2.
- Computes 128-bit $E_{dkey}(k_s) = k_s$ XOR ($dkey_2$ XOR r_{rx}) , where r_{rx} is XORed with the least-significant 64-bits of $dkey_2$.
- Sends SKE_Send_Eks message containing $E_{dkey}(k_s)$ and r_{iv} to the HDCP Receiver.

On receiving SKE_Send_Eks message, the HDCP Receiver

- Performs key derivation as explained in Section 2.8 to generate 128-bit $dkey_2$ where $dkey_2$ is the derived key when ctr =2.
- Computes $k_s = E_{dkey}(k_s)$ XOR ($dkey_2$ XOR r_{rx})

2.5 Authentication with Repeaters

Figure 2.5 illustrates 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 is implemented in parallel with the flow of encrypted content and Link Synchronization. The Link Synchronization process is explained in Section 2.6. The authentication with repeaters stage assembles a list of all downstream *Receiver ID*s connected to the HDCP Repeater through a permitted connection tree, enabling revocation support upstream.

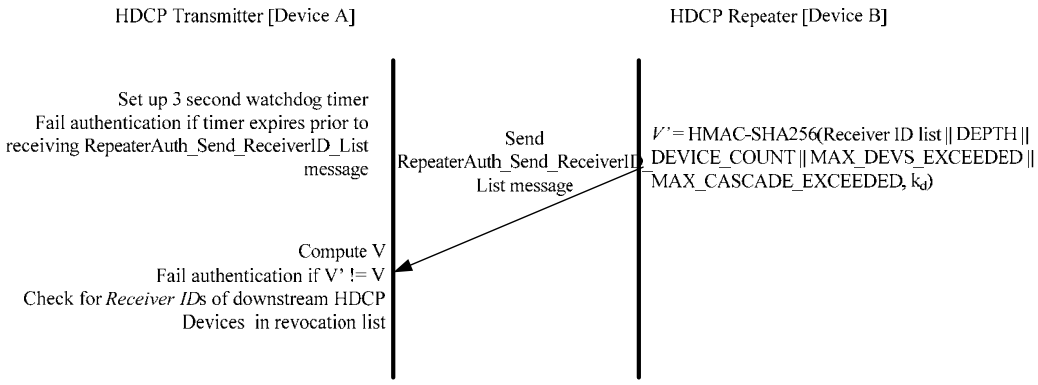


Figure 2.5. Authentication with Repeaters

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. 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 by computing *V* and checking this value against *V'* received as part of the RepeaterAuth_Send_ReceiverID_List message from the connected downstream HDCP Repeater. If *V* does not equal *V'*, the downstream Receiver ID list integrity check fails, and the HDCP Repeater must not send the RepeaterAuth_Send_ReceiverID_List message to the upstream HDCP Transmitter. Upstream HDCP Transmitters will detect this failure by the expiration of a watchdog timer set in the HDCP Transmitter. On expiration of the watchdog timer, authentication fails, the authentication protocol must be aborted and HDCP Encryption must be disabled (See Section 2.7 on handling authentication failures).

When the HDCP Repeater has assembled the complete list of connected HDCP Devices' *Receiver ID*s, it computes the 256-bit verification value *V'*.

V' = HMAC-SHA256(Receiver ID list || DEPTH || DEVICE_COUNT || MAX_DEVS_EXCEEDED || MAX_CASCADE_EXCEEDED, k_d)

HMAC-SHA256 is computed over the concatenation of Receiver ID list, DEPTH, DEVICE_COUNT, MAX_DEVS_EXCEEDED and MAX_CASCADE_EXCEEDED where

Receiver ID list is formed by appending downstream *Receiver ID*s in big-endian order. The key used for HMAC is k_d. When the Receiver ID list, *V'*, DEPTH and DEVICE_COUNT are available, the HDCP Repeater sends RepeaterAuth_Send_ReceiverID_List message to the upstream HDCP Transmitter.

The HDCP Transmitter, having determined that REPEATER received earlier in the protocol is 'true', sets a three-second watchdog timer. When the RepeaterAuth_Send_ReceiverID_List message is received, the HDCP Transmitter verifies the integrity of the Receiver ID list by computing *V* and comparing this value to *V'*. If *V* is not equal to *V'*, authentication fails, the authentication protocol is aborted and HDCP Encryption is disabled (See Section 2.7 on handling authentication failures).

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 (See Section 2.7 on handling authentication failures).

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 2.6, R1 has zero downstream HDCP Devices and reports a value of zero for both the DEPTH and the DEVICE_COUNT.

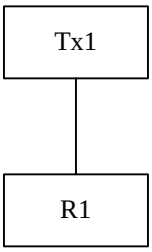


Figure 2.6. DEPTH and DEVICE_COUNT for HDCP Repeater

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.

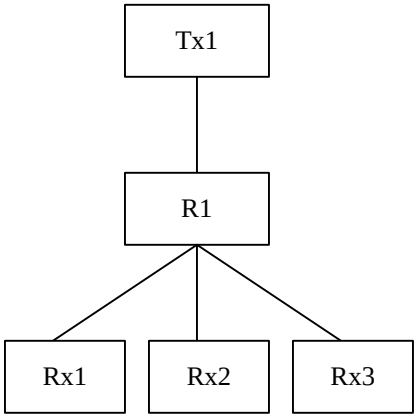


Figure 2.7. DEPTH and DEVICE_COUNT for HDCP Repeater

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

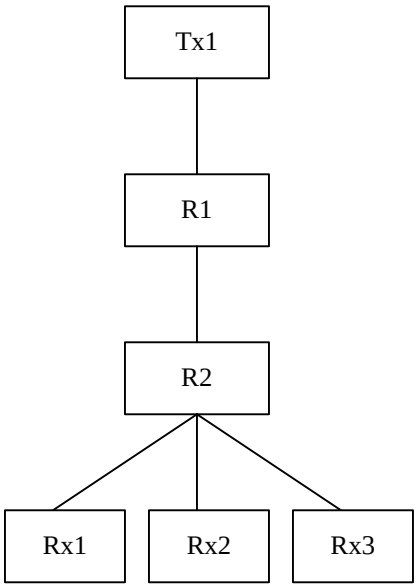


Figure 2.8. DEPTH and DEVICE_COUNT for HDCP Repeater

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'* and Receiver ID list.

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 (See Section 2.7 on handling authentication failures).

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 *fineInputCtr* and *coarseInputCtr* fields 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 *fineInputCtr* and *coarseInputCtr* in the header. (See Section 3.3 for details about *fineInputCtr* and *coarseInputCtr*). The HDCP Receiver updates its *fineInputCtr* and *coarseInputCtr* with the corresponding counter values received from the transmitter.

2.7 Authentication Failures

On an authentication failure at the HDCP Transmitter during the authentication protocol, the protocol must be aborted, if HDCP Encryption is enabled, it must be immediately disabled. Authentication must be reattempted at least once by the top-level HDCP Transmitter by the transmission of a new r_{tx} as part of the AKE_Init message. An exception to this rule is in the case of authentication failure due to failure of SRM integrity check or if the *Receiver ID* of an connected downstream HDCP Device is in the revocation list. Authentication need not be re-attempted in these cases. The HDCP Repeater initiates re-authentication on its downstream HDCP-protected interface ports only when it receives a re-authentication request i.e. a new r_{tx} value as part of the AKE_Init message, from upstream.

2.8 Key Derivation

Key derivation is illustrated in Figure 2.9.

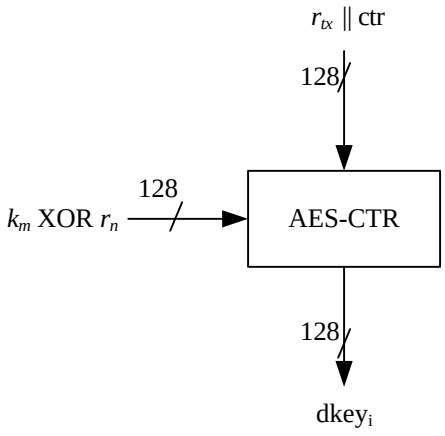


Figure 2.9. Key Derivation

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 HDCP 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 HDCP 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 $dkey_2$.

2.9 HDCP Transmitter State Diagram

As explained in Section 1.3, the HDCP Transmitter may support simultaneous connections to HDCP Receivers through one or more of its HDCP-protected interface ports. The HDCP Transmitter state diagram is implemented independently on each HDCP-protected interface port. The HDCP Transmitter can be considered to have two separate functions – a main HDCP Transmitter function and several HDCP Transmitter sub-functions. Each sub-function is associated with a specific HDCP-protected interface port on the transmitter and implements the HDCP Transmitter state diagram on the port. The main transmitter function ensures that the constraints on the HDCP System are met. This is explained further in Section 2.9.1.

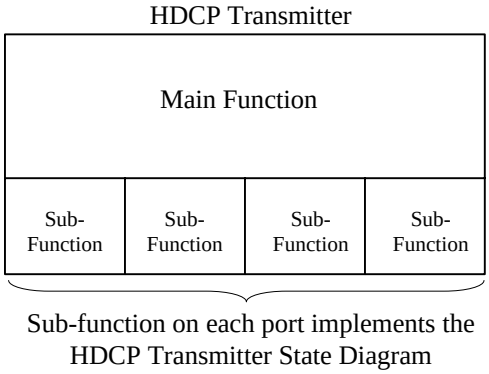
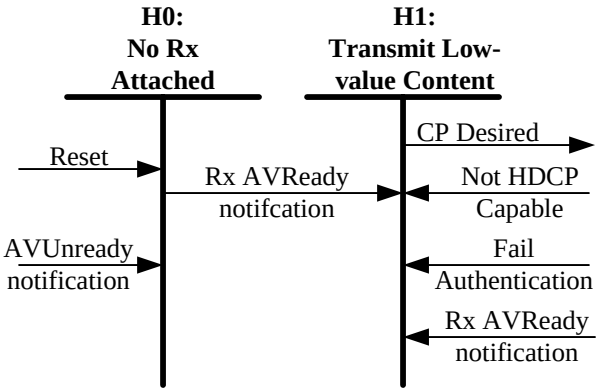


Figure 2.10. HDCP Transmitter Functions

The HDCP Transmitter Link State Diagram and HDCP Transmitter Authentication Protocol State Diagram (Figure 2.11 and Figure 2.12) 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 2.11.2.

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. An HDCP Receiver must be ready to authenticate immediately after going into AVReady state. In the event of 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 AVUnready notification or after a certain number of authentication re-attempts by the transmitter. During re-authentication attempts by the transmitter, HDCP Encryption is disabled (if it was previously enabled) and delivery of HDCP Content is terminated. HDCP Content delivery resumes only after successful re-authentication.

The transmitter must not initiate authentication unless the Setup and discovery procedures in the AVCL chapter of the WHDI specification, determine that the receiver is HDCP-capable. The Setup and discovery procedures in the AVCL chapter of the WHDI specification also indicate to the HDCP Transmitter the connect/disconnect status of the HDCP Receiver.



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

Figure 2.11. HDCP Transmitter Link State Diagram

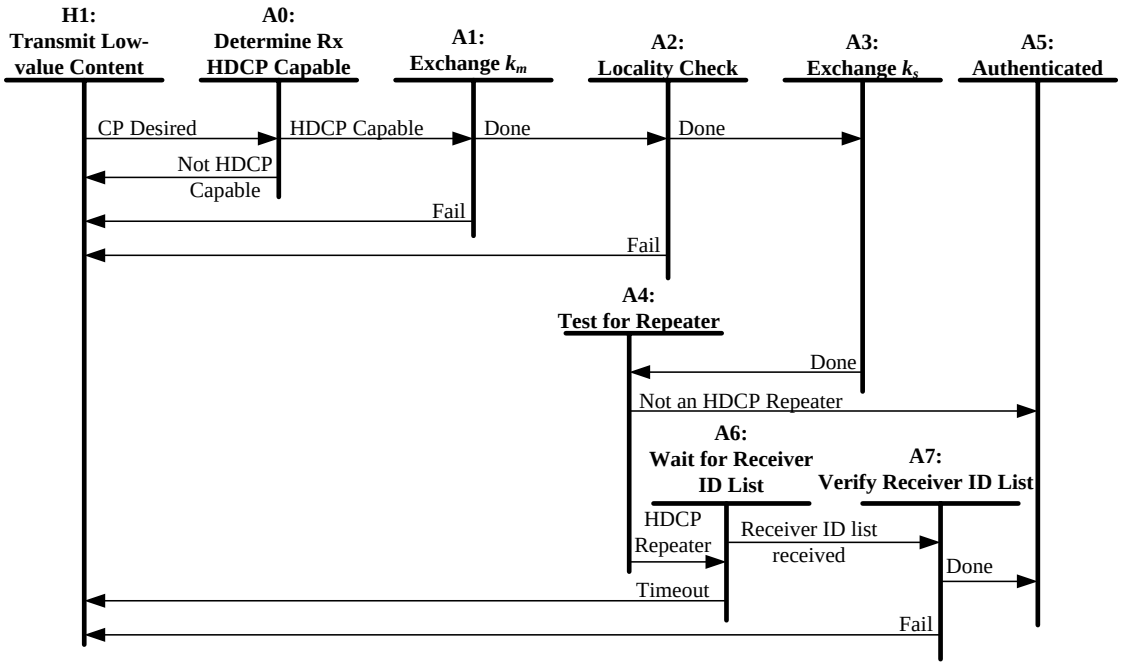


Figure 2.12. HDCP Transmitter Authentication Protocol State Diagram

Transition Any State:H0. Reset conditions at the HDCP Transmitter or disconnect of all HDCP capable receivers (all receivers in AVUnready state) cause the HDCP Transmitter to enter the No Receiver Attached state.

Transition H0:H1. The detection of a sink device (through AVReady notification) indicates to the transmitter that a sink device is ready for receiving HDCP content When the receiver is no longer active, the transmitter is notified through AVUnready notification.

State H1: Transmit Low-value Content. In this state the transmitter should begin sending an unencrypted signal with HDCP Encryption disabled. The transmitted signal can be a low value content or informative on-screen display. This will ensure that a valid video signal is displayed to

the user before and during authentication. At any time a Rx AVReady notification received from the connected 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 continues 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 $cert_{rx}$.

If the HDCP Transmitter does not have k_m stored corresponding to the *Receiver ID*, it generates $E_{kpub}(km)$ and sends $E_{kpub}(km)$ 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 $E_{kh}(k_m)$ 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 it 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, $E_{dkey}(k_s)$, 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 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. The watchdog timer is cleared. If both MAX_DEVS_EXCEEDED and MAX_CASCADE_EXCEEDED are not 'true', computes *V*, and verifies *V* == *V'*. The *Receiver ID*s from the Receiver ID list are compared against the current revocation list.

Transition A7:H1. This transition is made if *V* != *V'* or if any of the *Receiver ID*s in the Receiver ID list are found in the current revocation list. A MAX_CASCADE_EXCEEDED or MAX_DEVS_EXCEEDED error also causes this transition.

Transition A7:A5. This transition occurs if *V* == *V'*, none of the reported *Receiver ID*s are in the current revocation list, and the downstream topology does not exceed specified maximums.

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 undisrupted 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 3.6. However, the HDCP Transmitter must ensure that each connected HDCP Receiver receives distinct k_m and r_{tx} values.

2.9.1 Main HDCP Transmitter Function

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.

The first constraint is met by implementing the authentication protocol independently on each HDCP-protected interface port and verifying that the DEPTH and DEVICE_COUNT read from the connected repeater are less than or equal to 4 and 31 respectively (HDCP Transmitter sub-function). To meet the second constraint, the HDCP Transmitter (that is not an HDCP Repeater) performs an additional step after all its HDCP-protected interface ports have reached the terminal states of the authentication protocol i.e. State H0 (unconnected), State H1 (inactive or unauthenticated) and State A5 (authenticated). This is the main HDCP Transmitter function. For each of its HDCP-protected interface ports connected to an HDCP Repeater that have reached the authenticated state, State A5 and that will transmit the content stream received from a specific instance of the Upstream Content Control Function, the HDCP Transmitter computes the total number of HDCP Devices connected to each HDCP-protected interface port by incrementing the received DEVICE_COUNT on those ports by one (to account for the connected HDCP Repeater).

Total_Port_Device_Count = DEVICE_COUNT + 1

It then computes the total number of HDCP Devices connected to the HDCP Transmitter as follows

Total_Transmitter_Device_Count = $Total\_Port\_Device\_Count_1$ + ….. + $Total\_Port\_Device\_Count_n$, where n is the total number of HDCP-protected interface ports on the transmitter.

If the computed Total_Transmitter_Device_Count exceeds 32, the top-level HDCP Transmitter disables encryption and aborts the HDCP Session on all its HDCP-protected interface ports. The state diagram (Figure 2.13) and the description below relates to the main HDCP Transmitter function.

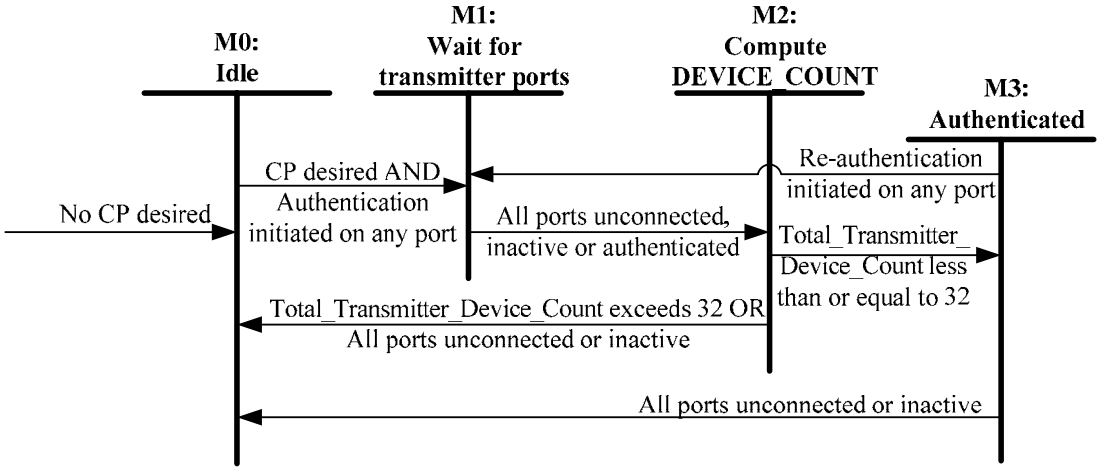


Figure 2.13. Main HDCP Transmitter Function State Diagram

Transition Any State:M0. The HDCP Transmitter transitions in to the Idle state when content protection is not desired by the upstream content control function. HDCP Encryption is disabled in this state.

Transition M0:M1. The transition occurs when content protection is desired by the upstream content control function and authentication has been initiated by the HDCP Transmitter on any of its HDCP-protected Interface ports by transmission of an AKE_Init message.

State M1: Wait for transmitter ports. In this state the transmitter waits for all its HDCP-protected interface ports to transition to the unconnected (State H0), inactive (State H1) or authenticated state (State A5).

Transition M1:M2. This transition occurs when all ports have transitioned to the unconnected, inactive or authenticated states.

State M2: Compute DEVICE_COUNT. The HDCP Transmitter computes the total number of HDCP Devices connected to it i.e. the Total_Transmitter_Device_Count.

Transition M2:M0. This transition occurs if the computed Total_Transmitter_Device_Count exceeds 32 or all transmitter ports have transitioned to unconnected or inactive state.

Transition M2:M3. This transition occurs if the computed Total_Transmitter_Device_Count for the HDCP Transmitter is less than or equal to 32.

State M3: Authenticated. At this time, and at no time prior, the HDCP Transmitter makes available to the Upstream Content Control Function upon request, information that indicates that the HDCP System is fully engaged and able to deliver HDCP Content, which means (a) HDCP Encryption is operational on each downstream HDCP-protected Interface Port connected to an HDCP Receiver, (b) processing of valid received SRMs, if any, has occurred, as defined in this Specification, and (c) there are no HDCP Receivers on HDCP-protected Interface Ports, or downstream, with *Receiver ID*s in the current revocation list.

Transition M3:M1. This transition occurs when re-authentication has been initiated by the HDCP Transmitter on any of its HDCP-protected Interface ports by transmission of an AKE_Init message.

2.10 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 2.14. For HDCP Repeaters, the upstream (HDCP Receiver) side is covered in Section 2.11.3.

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.

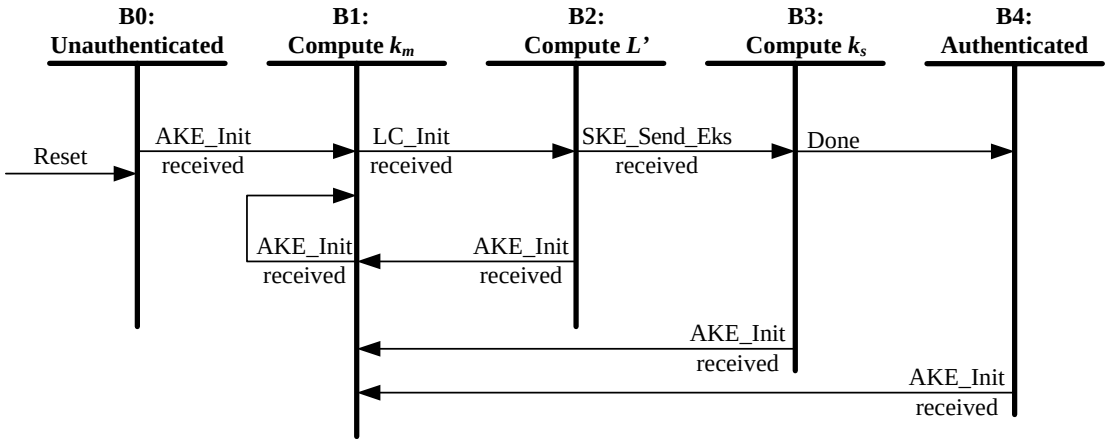


Figure 2.14. HDCP Receiver Authentication Protocol State Diagram

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, generates and sends r_{rx} as part of AKE_Send_rrx message. If AKE_No_Stored_km is received, it decrypts k_m with $kpriv_{rx,}$ 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 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 *fineInputCtr* and *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.

2.11 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 2.16 and Figure 2.17) is substantially the same as that for the host HDCP Transmitter (Section 2.9), with two exceptions. First, the HDCP Repeater is not required to check for downstream Receiver IDs in a revocation list. Second, the HDCP Repeater initiates authentication downstream when it receives an authentication request from upstream, rather than at detection of an HDCP Receiver on the downstream HDCP-protected Interface Port.

The HDCP Repeater 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. Similarly, when re-authentication is attempted by the upstream transmitter by sending a new r_{tx}, the HDCP Repeater immediately initiates 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.

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.

2.11.1 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.

AVUnready Notification. When an authenticated HDCP Receiver connected to the downstream HDCP Repeater connection is disconnected, the resulting AVUnready notification must not be propagated by the repeater to the upstream HDCP Transmitter when HDCP Content is flowing. The AVUnready notification must be propagated to the upstream HDCP Transmitter once the flow of HDCP Content stops or if there are no more authenticated HDCP Receivers connected to the HDCP Repeater.

AVReady Notification when HDCP Receiver is Re-connected. When an authenticated HDCP Receiver is disconnected and reconnected to the downstream port of the HDCP Repeater i.e. the downstream port of the repeater detects the same Receiver ID, and there were no intervening re-authentication requests from the upstream HDCP Transmitter during the time the HDCP Receiver was disconnected, the HDCP Repeater need not propagate the AVReady Notification to the upstream HDCP Transmitter. The HDCP Repeater may initiate authentication, complete the authentication protocol with the connected HDCP Receiver and enable HDCP Encryption.

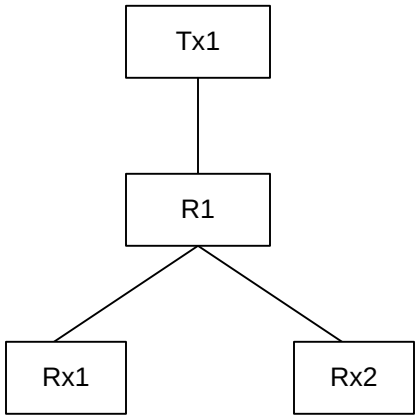
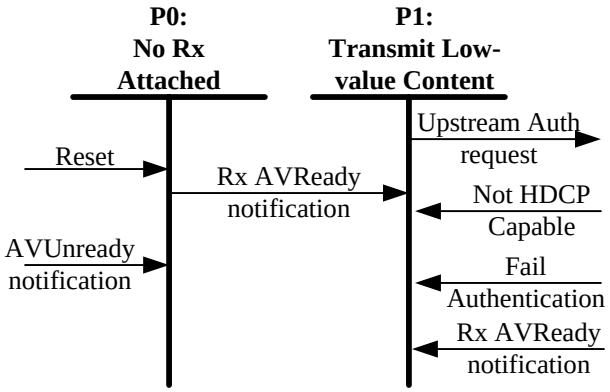


Figure 2.15. HDCP Receiver Reconnect

In Figure 2.15, Rx1 and Rx2 are authenticated HDCP Receivers connected to HDCP Repeater R1. When Rx2 is disconnected and reconnected and there were no intervening re-authentication requests from Tx1, R1 may authenticate Rx2 without propagating the AVReady Notification to Tx1.

2.11.2 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 2.16. HDCP Repeater Downstream Link State Diagram

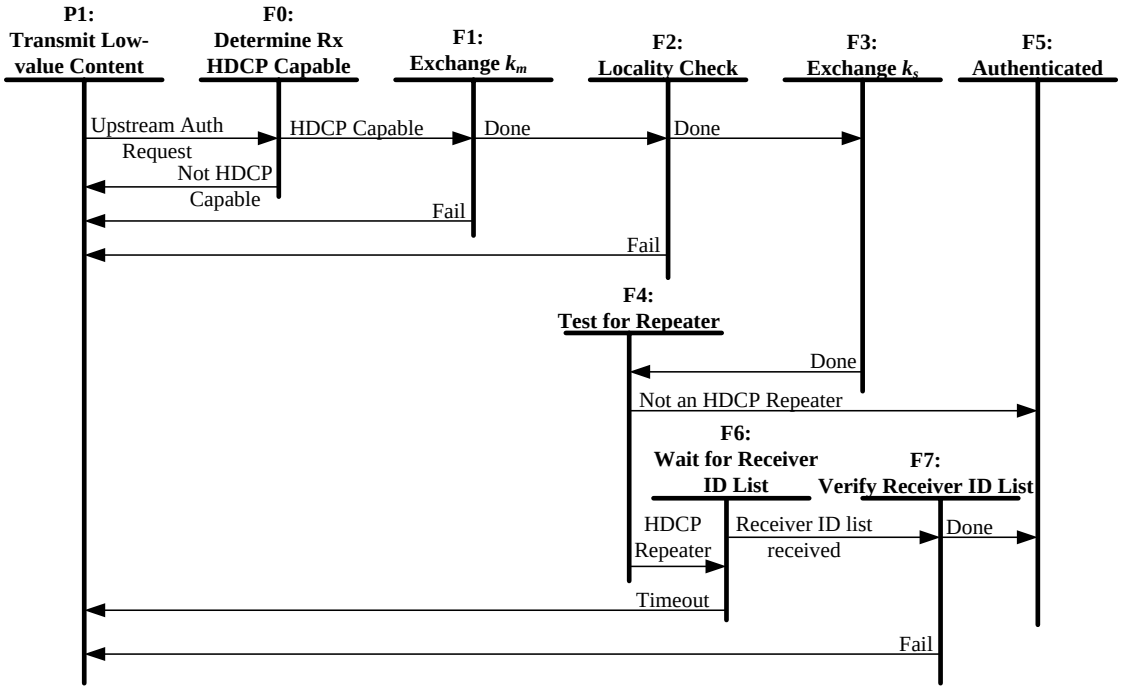


Figure 2.17. HDCP Repeater Downstream Authentication Protocol State Diagram

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 Repeater causes the downstream side to transition in to this state. From this state, the downstream side initiates authentication only when an Upstream Authentication Request is received i.e. the upstream side receives AKE_Init from the upstream HDCP Transmitter.

Note: As explained in Section 2.11.1, if a previously authenticated HDCP Receiver is re-connected and there were no intervening re-authentication requests from the upstream HDCP Transmitter during the time the HDCP Receiver was disconnected, the downstream side may initiate authentication with the HDCP Receiver without waiting for an Upstream Authentication Request.

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.

Transition F0:P1. If the receiver is not HDCP capable, the downstream side continues to transmit low value content or informative on-screen display 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. It receives AKE_Send_Cert from the receiver containing REPEATER and $cert_{rx}$.

If the downstream side does not have k_m stored corresponding to the *Receiver ID*, it generates $E_{kpub}(km)$ and sends $E_{kpub}(km)$ 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 it 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 three consecutive locality check failures. Locality check fails when *L'* is not received within 7 ms and the watchdog timer at the downstream side expires or on a mismatch between L and *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. The watchdog timer is cleared. If both MAX_DEVS_EXCEEDED and MAX_CASCADE_EXCEEDED are not 'true', computes *V*, and verifies *V* == *V'*. The *Receiver ID*s 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 *V* != *V'*. A MAX_CASCADE_EXCEEDED or MAX_DEVS_EXCEEDED error also causes this transition.

Transition F7:F5. This transition is made if *V* == *V'*, the downstream topology does not exceed specified maximums.

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 undisrupted operation.

2.11.3 HDCP Repeater Upstream State Diagram

The HDCP Repeater upstream state diagram, illustrated in Figure 2.18, 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.

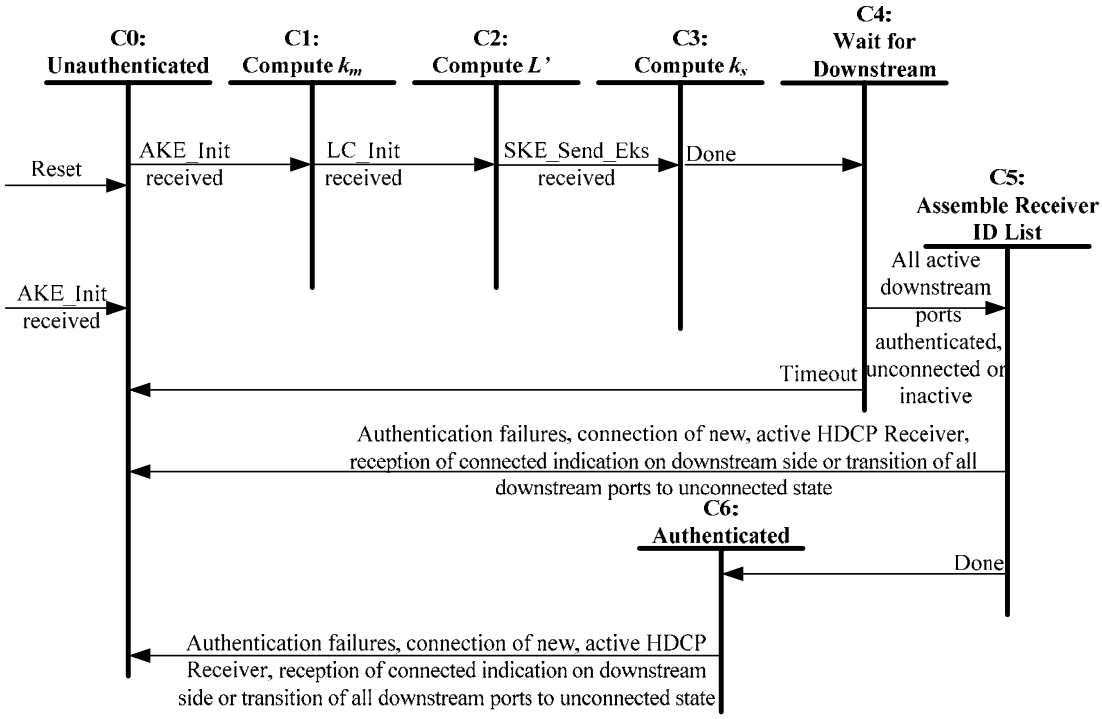


Figure 2.18. HDCP Repeater Upstream Authentication Protocol State Diagram

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_{tx} from the HDCP Transmitter to trigger the authentication protocol.

When the upstream side becomes unauthenticated due to any downstream HDCP-protected interface port transitioning to the unauthenticated state as a result of authentication failures, connection of a new, active HDCP Receiver on any downstream HDCP-protected interface port that previously did not have an active HDCP Receiver connected or reception of AVReady notification on the downstream side from the connected HDCP Repeater, it propagates the AVReady notification to the upstream HDCP Transmitter. Authentication failures are indicated by Transition F1:P1, Transition F2:P1, Transition F6:P1 and Transition F7:P1.

If a previously authenticated HDCP Receiver connected to the downstream HDCP-protected interface port is re-connected and there were no intervening re-authentication requests from the upstream HDCP Transmitter during the time the HDCP Receiver was disconnected, the upstream side need not transition to the unauthenticated state. The downstream side may authenticate the connected HDCP Receiver, as explained in Section 2.11.1.

When all downstream HDCP-protected interface ports transition to the unconnected state, the upstream becomes unauthenticated and propagates the resulting AVUnready notification to the upstream HDCP Transmitter.

Transition C0:C1. r_{tx} 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, generates and sends r_{rx} as part of AKE_Send_rrx message. If AKE_No_Stored_km is received, it decrypts k_m with $kpriv_{rx,}$ 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 AKE_Stored_km is received, the upstream side 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 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 $E_{dkey}(k_s)$ 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:C0. The watchdog timer expires before all downstream HDCP-protected Interface Ports enter the authenticated, unconnected or inactive state.

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 inactive 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.

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'* and Receiver ID list. 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.

Transition C5:C0. All authentication failures on the downstream side, connection of an active HDCP Receiver on the downstream HDCP-protected interface port that previously did not have an active downstream HDCP Receiver connected, reception of AVReady notification on the downstream side from the connected HDCP Repeater or transition of all downstream ports to the unconnected state cause this transition. This transition also occurs when topology maximums are exceeded. Authentication failures are indicated by Transition F1:P1, Transition F2:P1, Transition F6:P1 and Transition F7:P1.

If a previously authenticated HDCP Receiver connected to the downstream HDCP-protected interface port is re-connected and there were no intervening re-authentication requests from the upstream HDCP Transmitter during the time the HDCP Receiver was disconnected, the upstream side need not transition to the unauthenticated state. The downstream side may authenticate the connected HDCP Receiver, as explained in Section 2.11.1.

Transition C5:C6. RepeaterAuth_Send_ReceiverID_List message has been sent to the upstream HDCP Transmitter and topology maximums are not exceeded.

State C6: Authenticated. The upstream side has completed the authentication protocol. Periodically, it updates its *fineInputCtr* and *coarseInputCtr* with the corresponding counter values received from the transmitter.

Transition C6:C0. All authentication failures on the downstream side, connection of an active HDCP Receiver on the downstream HDCP-protected interface port that previously did not have an active downstream HDCP Receiver connected, reception of AVReady notification on the downstream side from the connected HDCP Repeater or transition of all downstream ports to the unconnected state cause this transition. Authentication failures are indicated by Transition F1:P1, Transition F2:P1, Transition F6:P1 and Transition F7:P1.

If a previously authenticated HDCP Receiver connected to the downstream HDCP-protected interface port is re-connected and there were no intervening re-authentication requests from the upstream HDCP Transmitter during the time the HDCP Receiver was disconnected, the upstream side need not transition to the unauthenticated state. The downstream side may authenticate the connected HDCP Receiver, as explained in Section 2.11.1.

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 C6) must be implemented asynchronously from the rest of the state diagram. The transition into State C6 must occur from any state for which encryption is currently enabled. Also, the transition from state C6 returns to the appropriate state to allow for undisrupted operation.

2.12 Converters

2.12.1 HDCP 2 – HDCP 1.x Converters

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

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

Note: Locality check is not implemented in the downstream HDCP-protected interface ports.

The HDCP 2 compliant upstream side implements the state diagram as explained in Section 2.11.3 with these modifications.

- **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 the unconnected state or the inactive state does not add to the list. A downstream HDCP-protected Interface Port that arrives in an authenticated state that has an HDCP Receiver that is not an HDCP Repeater connected, adds the *Bksv* of the connected HDCP Receiver to the Receiver ID list. Downstream HDCP-protected Interface Ports that arrive in an authenticated state that have an HDCP Repeater connected will cause the KSV list read from the connected HDCP Repeater, plus the *Bksv* of the connected HDCP Repeater itself, to be added to the list. KSVs are used in place of *Receiver ID*s and are added to the Receiver ID list in big-endian order

  When the Receiver ID list (comprising KSVs of connected downstream HDCP 1.x Receivers, where the KSVs are added to the list in big-endian order) 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'* and Receiver ID list. 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.

2.12.2 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.

The HDCP 1.x compliant upstream side implements the state diagram explained in the corresponding HDCP 1.x specification (See Section 1.5). When any downstream HDCP-protected interface port transitions to the unauthenticated state as a result of authentication failures or connection of a new, active HDCP Receiver, the upstream side becomes unauthenticated.

The HDCP 2 compliant downstream side implements the state diagram as explained in Section 2.11.2 with these modifications.

- **State F7: Verify Receiver ID List**. The watchdog timer is cleared. If both MAX_DEVS_EXCEEDED and MAX_CASCADE_EXCEEDED are not 'true', computes *V*, and verifies *V* == *V'*. The *Receiver ID*s 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 ID*s in little-endian order. The upstream HDCP Transmitter must be informed if topology maximums are exceeded.

If 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.

2.13 Session Key Validity

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

If HDCP Encryption was disabled, from its enabled state, due to the detection of AVReady notification, AVUnready notification or authentication failures, the session key expires. The most upstream HDCP Transmitter initiates re-authentication by the transmission of a new r_{tx}. 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 session key does not expire. The HDCP Transmitter maintains the encryption parameters used during the HDCP Session i.e. *fineInputCtr* and *coarseInputCtr* values 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 *fineInputCtr* and *coarseInputCtr* with the corresponding counter values received from the transmitter. (See Section 2.6 on Link Synchronization).

2.14 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.

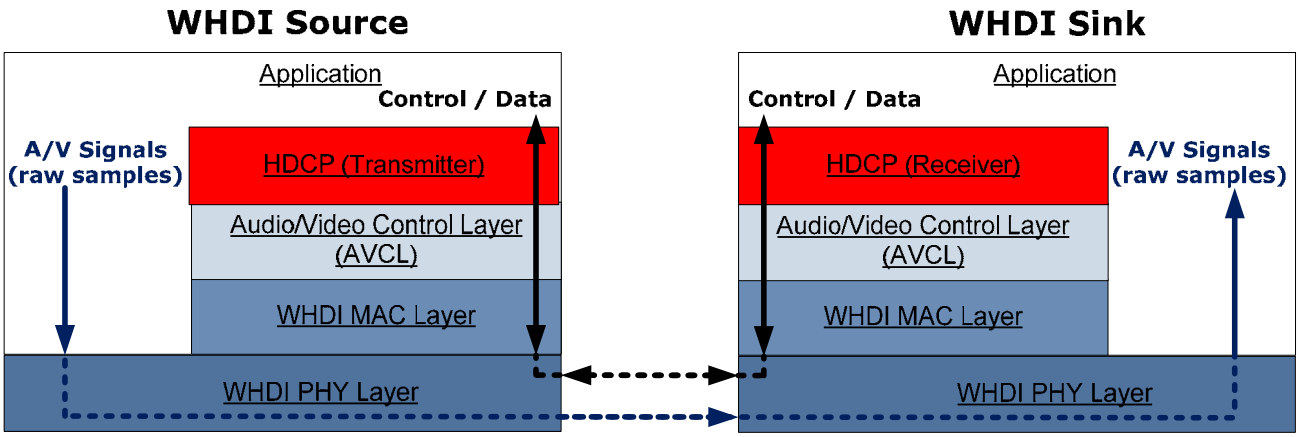


Figure 3.1. WHDI with HDCP Block Diagram (Informative)

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 date 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

WHDI uses different HDCP Cipher structures for encryption of coarse data and fine data.

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.

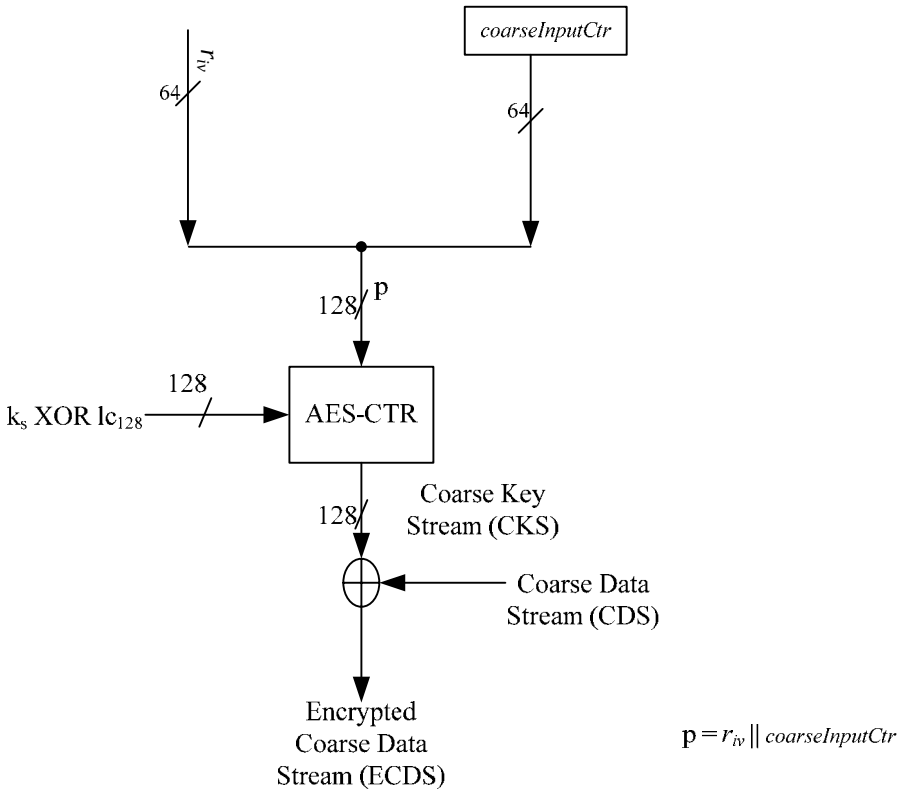


Figure 3.2. HDCP Cipher Structure for Coarse Data

k_s is the 128-bit session key which is XORed with lc_{128}.

$p = r_{iv} \| coarseInputCtr$. 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.

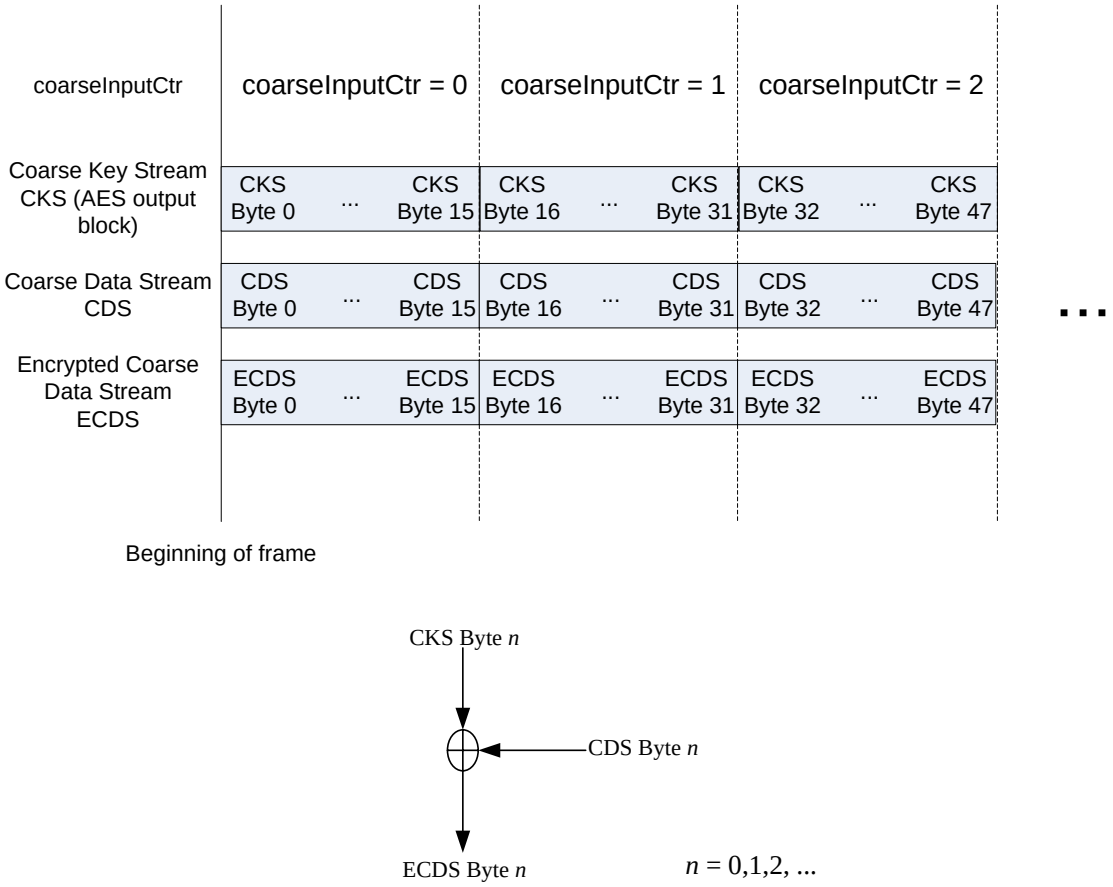


Figure 3.3. Byte Ordering in Coarse Data Encryption

3.3.2 HDCP Cipher for Fine Stream

The HDCP Cipher for fine data consists of a 128-bit AES module that is operated in a Counter (CTR) mode as illustrated in Figure 3.4.

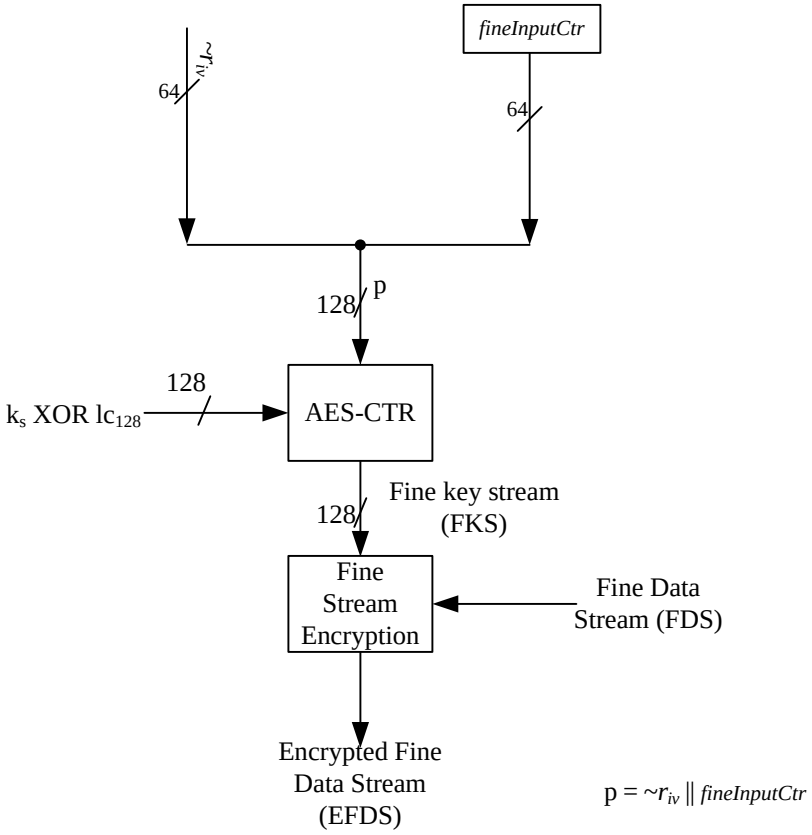


Figure 3.4. HDCP Cipher Structure for Fine Data

k_s is the 128-bit session key which is XORed with lc_{128}.

p = ~ r_{iv} || *fineInputCtr*, where ~ r_{iv} is the bitwise inversion of r_{iv}. All values are in big-endian order.

fineInputCtr 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 all WHDI fine payload data.

During HDCP Encryption, the key stream produced by the AES-CTR module is used to encrypt the fine data stream. *fineInputCtr* is incremented by one following the generation of every 128-bit block of fine key stream. The value of *fineInputCtr* must never be reused for a given set of encryption parameters i.e. k_s, r_{iv}.

The fine stream encryption is composed of permutation and sign inversion, as follows. The output of the AES-CTR block is assembled into a long fine key stream, denoted by FKS[n], where n is the byte index, and least significant bytes have lower values of n. The key stream is divided into blocks of 33 bytes – $FKS_b[k]$, where b is the block index and k =0..32. The fine data stream is divided into blocks of 256 complex samples – $FDS_b[i]$, where b is the block index and i = 0..255. Each block of 33 bytes of key stream is used to encrypt a block of 256 fine data samples. The first byte of the key stream in each block, denoted by $P_b = FKS_b[0]$, is used to permute the fine samples within the 256 samples block.. The first block index, b, is 1, and P_0=0. The permuted block of the fine data stream $PFDS_b[i]$, is constructed as follows:

$$PFDS_b[\, i \,] = FDS_b[\, i \ \text{XOR} \ P_{b-1} \ \text{XOR} \ P_b] \ ; \text{where } i = 0..255$$

Then, the remaining 32 bytes of each key stream block determine the sign inversion of the corresponding fine data sample. Each byte determines the sign of 8 fine samples:

$$EFDS_b[\,i\,] = \left(PFDS_b[\,i\,] * ((-1) \wedge B_j) \right) \text{ ; where } i = 0..255,\ j = (i \bmod 8)$$

B_j is the (i mod 8) bit in the $FKS_b[\lfloor i/8 \rfloor + 1]$ byte, and $EFDS_b[i]$ is the encrypted fine stream block. These blocks are then concatenated, from lowest index to highest one, to produce the encrypted fine stream.

The 256 samples encryption block boundary must be aligned with the start of frame data. The last block in an encrypted frame is also 256 samples long.

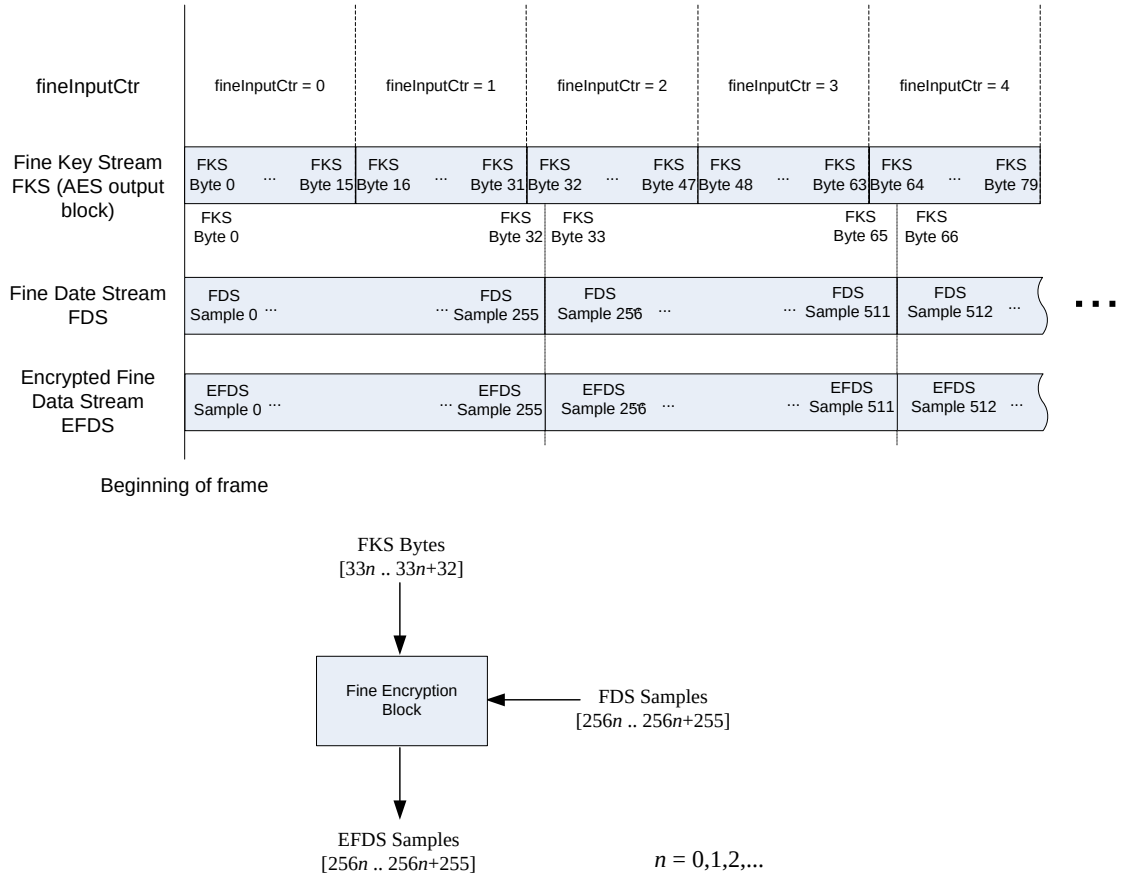


Figure 3.5. Block Division and Sample Ordering for Fine Data Encryption

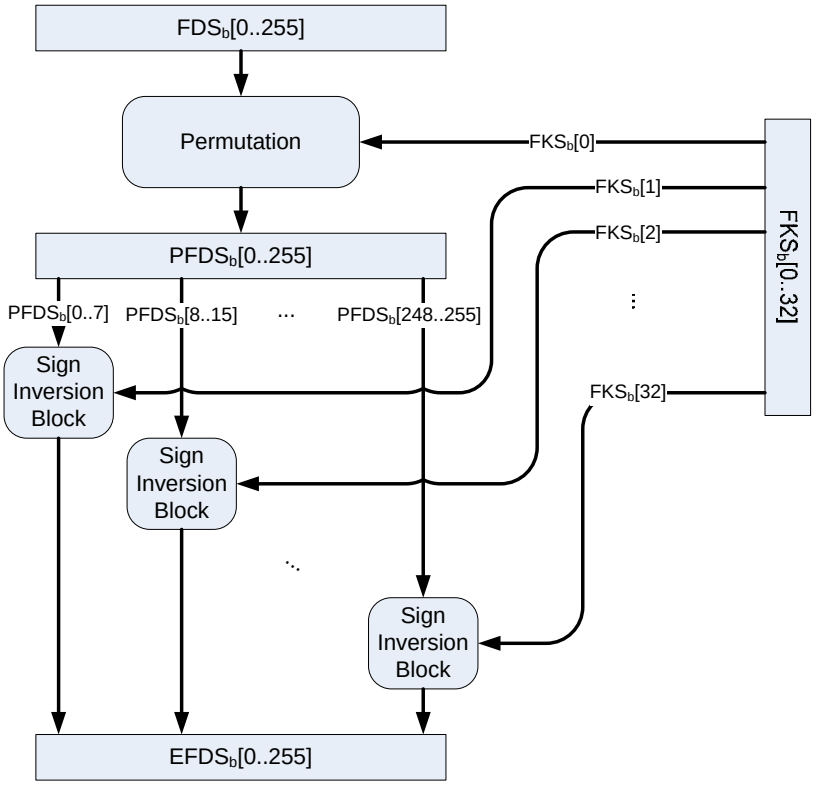


Figure 3.6. Fine Encryption Block

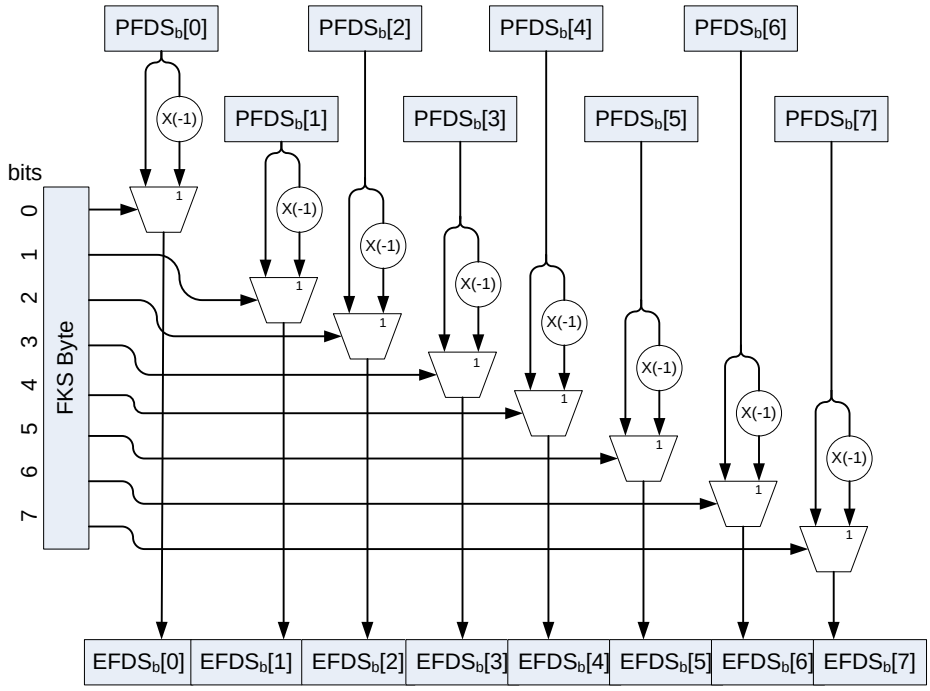


Figure 3.7. Sign Inversion Block

3.4 HDCP Encryption Indication

Any WHDI frame containing an HDCP encrypted payload must include the 64 bit *coarseInputCtr* field and the 64 bit *fineInputCtr* 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* and *fineInputCtr* fields are not included in the WHDI Extended Header.

The *coarseInputCtr* and *fineInputCtr* are loaded at the beginning of each frame with the *coarseInputCtr* and *fineInputCtr* values 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. Similarly, the *fineInputCtr* value transmitted in the frame header is used to generate the first 16 bytes fine key stream for that frame, used (together with the next 17 bytes of key stream) in the encryption of the first 256 samples of the fine data payload for that frame.

3.5 HDCP Cipher Block

The HDCP Cipher module used for coarse data together with the HDCP Cipher module used for fine data is referred to as HDCP Cipher Block.

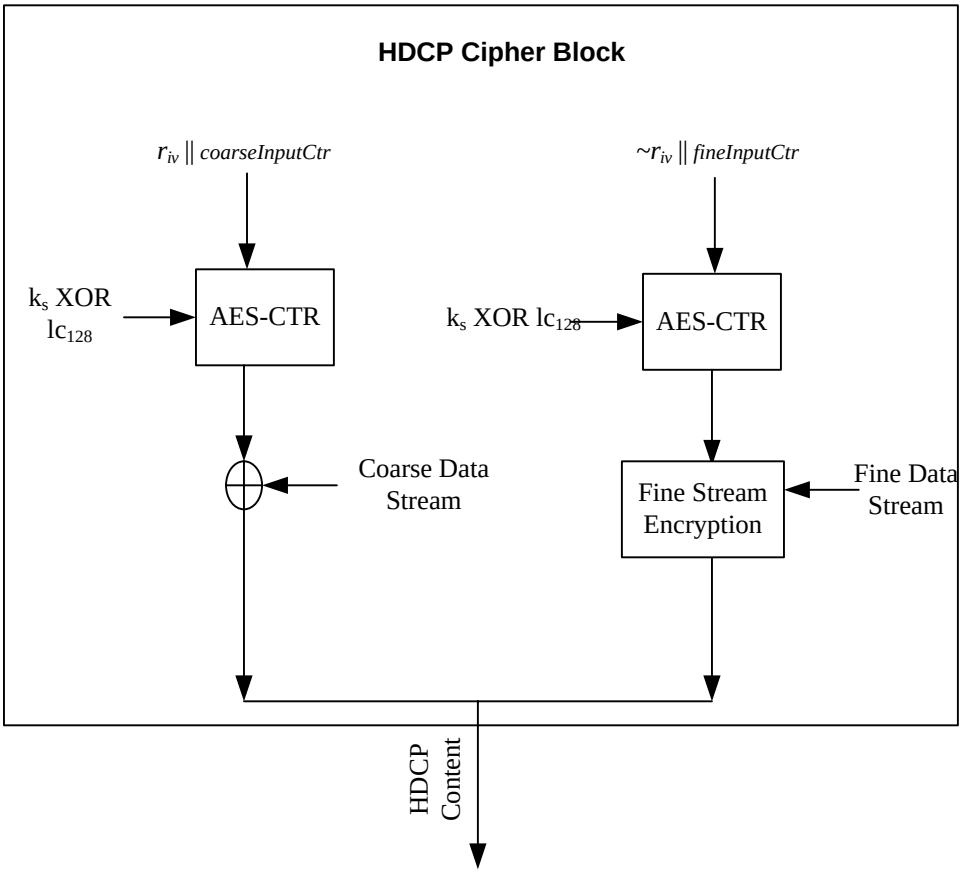


Figure 3.8. HDCP Cipher Block

3.6 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_{tx} 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_{tx} value.

As illustrated in Figure 3.9, 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.

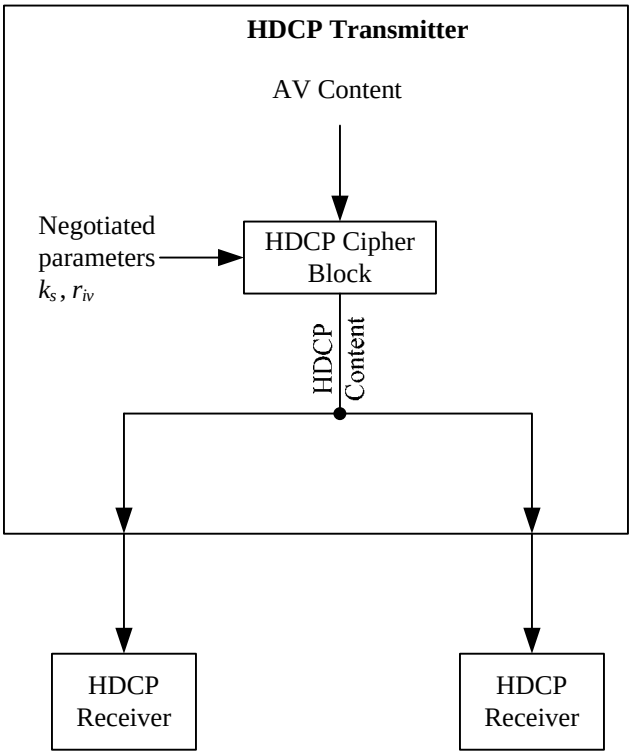


Figure 3.9. k_s and r_{iv} Shared across HDCP Links

As illustrated in Figure 3.10, 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.

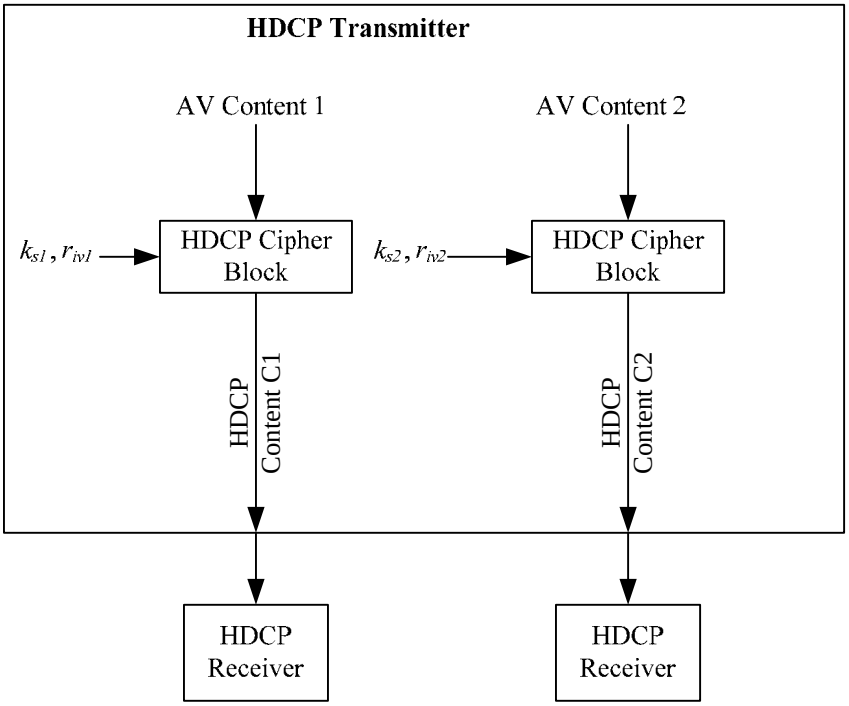


Figure 3.10. Unique k_s and r_{iv} across HDCP Links

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
Reserved	13-31

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.

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_{tx}	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
$E_{kpub}\_k_m$	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)

Field	No. of Bytes	Identifier
msg_id	1	uint
L'	32	uint

Table 4.11. LC_Send_L_prime Payload

4.2.10 SKE_Send_Eks (Transmitter to Receiver)

Field	No. of Bytes	Identifier
msg_id	1	uint
$E_{dkey\_}k_s$	16	uint
r_{iv}	8	uint

Table 4.12. SKE_Send_Eks Payload

4.2.11 RepeaterAuth_Send_ReceiverID_List (Receiver to Transmitter)

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

Receiver ID list = *Receiver ID*$_0$ || *Receiver ID*$_1$ || … || *Receiver ID*$_n$, 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'

Field	No. of Bytes	Identifier
msg_id	1	uint
MAX_DEVS_EXCEEDED	1	Bool
MAX_CASCADE_EXCEEDED	1	bool
DEVICE_COUNT	1	Uint
DEPTH	1	uint
V'	32	uint
Receiver_ID$_1$	5	uint
Receiver_ID$_2$	5	Uint
…		
Receiver_ID$_n$, where n = DEVICE_COUNT	5	Uint

Table 4.13. RepeaterAuth_Send_ReceiverID_List Payload

Note:

- Fields *V'*, Receiver ID list (*Receiver_ID*$_1$, …., *Receiver_ID*$_n$) must not be transmitted if MAX_DEVS_EXCEEDED=true or MAX_CASCADE_EXCEEDED=true
- The length of this message is variable, dependent on the DEVICE_COUNT

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 $cert_{rx}$. 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, $kpriv_{rx,}$ 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

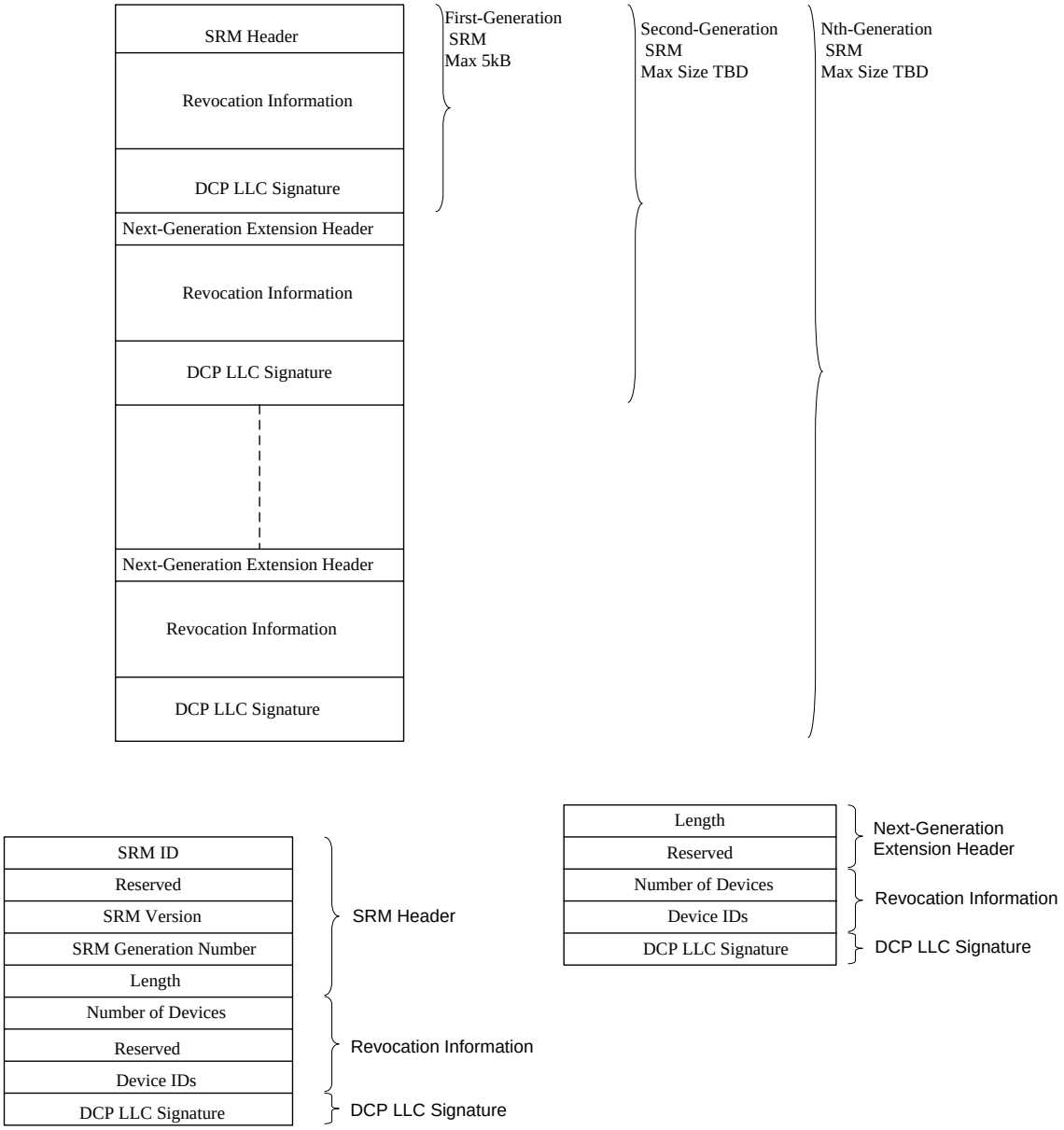


Figure 5.1. SRM Generational Format

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[±]?	Integrity Required[±]?
lc_{128}	Yes	Yes
$kpub_{dcp}$	No	Yes
$cert_{rx}$	No	No
$kpub_{rx}$	No	Yes
Receiver ID	No	Yes
$kpriv_{rx}$	Yes	Yes
r_{tx}	No	Yes[*]
r_{iv}	No	Yes[*]
REPEATER	No	Yes
r_{rx}	No	Yes[**]
k_m	Yes	Yes[*]
k_d	Yes	Yes[*]
$dkey_i$	Yes	Yes[*]
H	Yes	Yes
H'	No	No
m	No	No
k_h	Yes	Yes
r_n	No	Yes[*]

[±] According to the robustness rules in the HDCP Adopter's License

[*] Only within the transmitter
[*] Only within the transmitter
[**] Only within the receiver

L	Yes	Yes
L'	No	No
k_s	Yes	Yes*
V	Yes	Yes
V'	No	No
Receiver ID list	No	Yes
DEPTH	No	Yes
DEVICE_COUNT	No	Yes
MAX_DEVS_EXCEEDED	No	Yes
MAX_CASCADE_EXCEEDED	No	Yes
coarseInputCtr	No	Yes*
fineInputCtr	No	Yes*

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