

February 20, 2008

The nuts and bolts of HDCP

By Robert Carter, Crestron, Inc

Introduction

As the digital distribution of television, movies and music expands, content providers are growing increasingly concerned about the simplicity with which content pirates can [copy](#) and share copyrighted material. Various [digital rights management](#) (DRM) schemes have been developed to ensure that television shows, movies and music can only be viewed by authorized parties (i.e., paying consumers). The Content Scrambling System (CSS) used to encrypt DVDs and Apple's Fairplay technology are two widely-known [DRM](#) schemes.

To protect digital content as it's transmitted over cables between devices, [Intel](#) developed a DRM scheme known as the High-bandwidth Digital Content Protection system (HDCP). Intel then handed licensing responsibilities over to its subsidiary, Digital Content Protection, LLC (DCP). HDCP has recently enjoyed rapid, widespread adoption in the consumer electronics space, but has been plagued by interoperability issues. This article provides a technical interpretation of the HDCP specification and then explores the reasoning behind some of the problems.

HDMI

HDCP was originally designed to protect AV content transmitted over the Digital Video Interface (DVI), then the High Definition Multimedia Interface (HDMI). Recently, DCP added HDCP support for DisplayPort, a competing transmission interface. The remainder of this article will focus on HDCP as it applies to HDMI, but the same concepts apply with [DVI](#) and DisplayPort.

The HDMI specification defines an interface for carrying digital audiovisual content from a source, like a [DVD](#) player, to a sink, or [display](#) device, such as a TV. Devices such as switches or A/V receivers may accept and re-transmit [HDMI](#) content, and are known as repeaters.

Repeater devices have two separate HDMI connections: the upstream connection with the source and the downstream connection with the sink (or another repeater). Within each connection, the upstream device sends audiovisual data to the downstream device.

The physical HDMI [cable](#) carries many signals:

- *TMDS*. The audiovisual data is encoded into three Transition Minimized Differential Signaling (TMDS) data channels. These channels and a TMDS [clock](#) are carried over four differential pairs from the source to the sink.
- *DDC*. The Digital Display Channel (DDC) is a [communications](#) interface similar to I2C. This interface provides two-way communication in a master-slave relationship. The upstream device is the DDC master and the downstream device is the DDC slave.
- *Hot Plug Detect*. The sink indicates its presence to the source with the Hot Plug Detect (HPD) signal. The sink can toggle the Hot Plug Detect signal to reset the HDMI connection (and thus, the HDCP session).
- *RxSense*. Though not specifically defined by HDMI, many devices support a feature known as RxSense. There is no RxSense wire, but rather, sources can detect that a sink has terminated the TMDS differential pairs. Similarly to HPD, this signal can be used to detect the presence of a sink.

The HDMI cable carries other signaling information that is beyond the scope of this article.

HDCP Overview

To protect content, the HDCP system first authenticates HDCP devices and then encrypts the content. [Authentication](#) occurs over the DDC channel and assures that all devices receiving the content are licensed and authorized to receive the content. After successful authentication, the [TMDS](#) data streams are encrypted to prevent other devices from eavesdropping on the content during transmission.

HDCP devices are organized in a tree topology, as shown in the figure below. The source is the root and sinks are the leaves, while repeaters make up the branches of the tree. The tree may have at most 127 devices and may be no more than 7 levels deep.

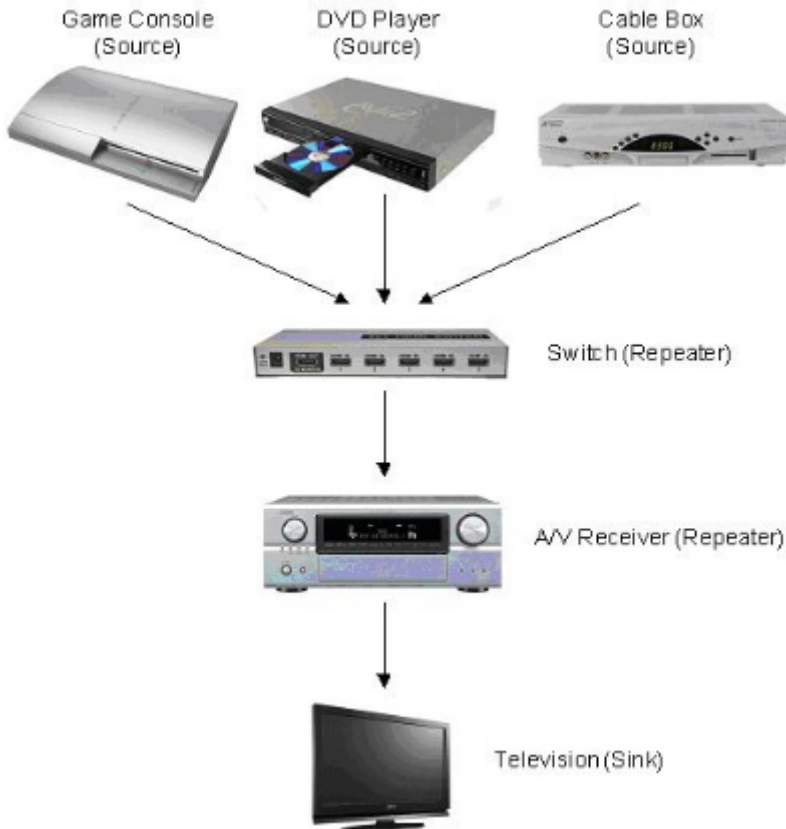


Figure 1, Example HDCP Tree

A single point-to-point HDCP [link](#) can only involve one HDCP transmitter and one HDCP receiver. As such, a repeater must decrypt the content at the HDCP receiver on each of its inputs. The repeater must then re-encrypt the data with an HDCP transmitter on each of its outputs. The [repeater](#) must inform the upstream device of its downstream connections, but it is the repeater's responsibility to maintain those connections.

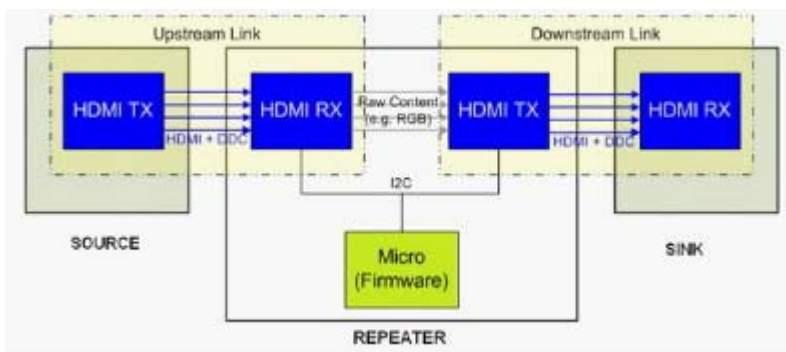


Figure 2, Repeater diagram

HDCP Implementation Elements

The HDCP specification doesn't go into implementation specifics, but, in practice, device manufacturers typically buy HDCP chips from a DCP-licensed silicon vendor. These chips usually also provide TMDS encoders or decoders and other HDMI-specific features. Every transmitting device will have at least one HDCP transmitter [chip](#) and every receiving device will have at least one HDCP receiver chip. The HDCP transmitters and receivers frequently require a microprocessor to implement the authentication state machines.

To support HDCP, each transmitter and receiver must possess the following elements:

- *Keys.* : Each HDCP transmitter and receiver has forty unique 56-bit private keys. These keys are provided by DCP to licensed HDCP chip vendors, who pre-load the keys onto the chips before selling them to device manufacturers. These keys must never leave the chip and may not be read by any other device.
- *KSVs.* : Each HDCP chip also has a public 40-bit value known as the Key Selection Vector (KSV). Each KSV consists of twenty binary 1s and twenty binary 0s. The KSVs and keys of all licensed HDCP devices are mathematically related according to a cryptographic key exchange [protocol](#) similar to Blom's scheme. In this scheme, any two licensed devices can swap KSVs and use them, along with their private keys, to come up with a shared secret key. This shared key can be used to encrypt and decrypt the TMDS stream. The KSV can also be used to uniquely identify a transmitter or receiver.
- *HDCP Cipher.* : Each chip must implement the HDCP Cipher. The cipher accepts a seed value and uses it to generate a deterministic pseudo-random stream of data. All HDCP Cipher implementations should generate the same [output](#) stream given the same seed value. This publicly defined cipher is used during both authentication and encryption.
- *Registers.* : Each HDCP receiver must provide a series of pre-defined DDC-accessible registers. All of the HDCP communications between the source and sink occurs by the source reading from and writing to these registers.

The Authentication and [Encryption](#) Protocols

HDCP authentication consists of three parts:

- *Part 1:* : The source authenticates with the device connected to its output. If successful, encryption is enabled and A/V content transmission begins.
- *Part 2:* : This part is only used if the downstream device is a repeater. The repeater authenticates with the devices connected to its output(s) and passes the HDCP tree [topology](#) information up to the source.
- *Part 3:* The source performs periodic checks with the downstream device to ensure that encryption is in sync.

HDCP also supports a key revocation mechanism that is designed to prevent content from being transmitted to known hacked devices. If any part of authentication fails or any revoked devices are found in the HDCP tree, the transmitter must stop sending protected content and authentication starts over at Part 1.

Authentication, encryption, and revocation are described in detail in the following sections.

Authentication Part 1

Part 1 of authentication is a key exchange protocol. The transmitter and receiver must calculate a common secret session key to be used for encryption. If they can't come up with the same key value, authentication fails and the receiver won't be able to decrypt the content.

The session key is derived from each device's private key according to the following protocol:

First the transmitter generates a random number A_n and sends it to the receiver. This value will be used later in the protocol. The devices then exchange KSVs. The receiver also sends its REPEATER bit, a flag that indicates whether or not it's part of a repeater.

Now each device has the other device's Key Selection Vector (KSV). Note that I spelled that out again: that name looks like it means something, doesn't it? Each device uses the other device's Key Selection Vector to select twenty of its own keys. The forty bits in the KSV correspond to the indexes of each of the forty private keys. For every set [bit](#) in the received KSV, the local private key at that index is selected. All KSVs have twenty set bits, so twenty keys are selected.

The devices then each add up their selected keys to come up with the sums K_m and $K_{m'}$, for the transmitter and receiver, respectively. For authentication to succeed, K_m and $K_{m'}$ must match. Think about that for a minute. Each device tells the other which of its own unique, secret keys to select, and they both come up with the same sum. That may seem counter-intuitive, but it's the aforementioned mathematical relationship between the keys and the KSVs that accounts for this behavior.

The source must determine whether K_m and $K_{m'}$ match. However, they are secret values, so they can't be transmitted over the DDC lines. We need some mechanism by which the transmitter can verify that those values match without sending them across the cable for everyone to see.

To accomplish this, each device feeds K_m (or $K_{m'}$), the random number A_n , and the REPEATER bit into their respective HDCP cipher engines. The resulting data stream is split into three values:

- R_0/R_0' : This return value may be shared between the devices and is used to verify that authentication was successful.
- $K_s/K_{s'}$: This value is kept private and is used as the encryption session key for the HDCP cipher.
- M_0/M_0' : This value is also kept private and is used in Part 2 of authentication (if the downstream device is a repeater).

The receiver sends R_0 to the transmitter, which compares it against its' own R_0 value. If they match, that proves that the sums K_m and $K_{m'}$ matched, and authentication is successful. Furthermore, the session keys K_s and $K_{s'}$ match, so the receiver will be able to decrypt the content encrypted by the transmitter.

If Part 1 of authentication was successful, the transmitter may begin sending encrypted content. If the downstream device is a repeater, the repeater must authenticate with its own downstream device according to the same protocol. The transmitter then starts a 5-second timer to allow for the repeater to perform Part 2 of authentication. If Part 2 fails or times out, authentication fails and the transmitter must stop transmitting the protected content.

Authentication Part 2

Part 2 of authentication only occurs if the downstream device is a repeater. The purpose of Part 2 is to inform the source of all downstream devices and the HDCP tree depth. The source uses this information to ensure that the tree topology maximums haven't been exceeded and to ensure that none of the downstream devices have been revoked by DCP.

The repeater first assembles a list of the KSVs of all downstream devices, as well as the device count and the tree depth. The repeater then passes this information up to the source. To ensure that this information hasn't been tampered with during transmission, each device takes this list, appends its secret value M_0/M_0' from Part 1, and calculates a SHA-1 hash of the whole thing. The transmitter reads the hash result from the receiver and compares it against its own. If they match, Part 2 of authentication is successful.

Authentication Part 3

All HDCP devices are considered authenticated after successful completion of Authentication Parts 1 and 2. Part 3 is simply a link [integrity](#) check to ensure that encryption is in sync between all transmitter/receiver pairs in the tree.

To support link integrity checks, the return values R_i and R_i' roll over to a new value every 128 frames. Recall that the initial R_i values R_0 and R_0' were generated during Part 1 of authentication. Every two seconds, the transmitter compares the receiver's R_i' value against its own R_i value to see if they match. If they don't, encryption is out of sync and the receiver can't correctly decrypt the content. The user will see a scrambled or "snowy" image on the screen. In this case the transmitter must restart authentication from the beginning.

Encryption

The transmitter uses the HDCP cipher engine to encrypt the content. The session key K_s from Authentication Part 1 is used as the seed value, and the cipher output stream is simply XORed with the audiovisual content. The transmitter then sends the XOR output to the receiver as the encrypted data stream.

To decrypt the data, the HDCP receiver seeds its own cipher engine with its matching value $K_{s'}$. Since K_s equals $K_{s'}$, the cipher output matches that of the transmitter. The receiver then XORs its cipher output stream with the encrypted data, and the output is the decrypted audiovisual content.

Revocation

The HDCP designers recognized that despite their best efforts, private device keys could possibly be compromised. If such a compromise were discovered, the designers wanted a method by which they could prevent content from being sent to the compromised units.

To this end, HDCP sources are required to manage System Renewability Messages (SRMs). SRMs carry a list of revoked KSVs and may be provided with the audiovisual content. For instance, a SRM could be stored on an HD-DVD or may be transmitted with a cable television signal.

If presented with an SRM, the HDCP source must check all the downstream KSVs obtained during Parts 1 and 2 against the KSVs listed in the SRM. If there are any matches, the HDCP source must stop transmitting the protected content.

SRMs are only supported by the HDCP source; neither repeaters nor sinks handle SRMs.

Interoperability Issues

Now that we understand the protocol, we can explore some of the issues that have been causing problems in the industry. Consumers have been plagued with flashing and snowy screens, long authentication times, disabled outputs, and complete failure. Many of the problems can be ascribed to the following causes:

Complexity. Gone are the days when a simple [analog](#) signal was fed through some amplifiers and up to your CRT. The HDCP protocol is managed by a microcontroller running complex state machines. Many manufacturers didn't initially respect this complexity when selecting parts and establishing development schedules.

Implementation differences. Manufacturers implemented some seemingly minor details in different ways. HPD and RxSense behavior, for instance, isn't well standardized. Some sinks will toggle one to reset HDCP, some the other. Some will toggle one then the other. Some sources will always authenticate twice when connected to a repeater. The list goes on. Small differences like this can wreak havoc on the HDCP state machines of attached devices and cause ugly [video](#) problems for the customer.

Requirement Confusion. There has been some confusion as to the legal requirements for HDCP. Some device manufacturers have taken the safe route and encrypted everything, regardless of whether or not the content required it. Some disable analog outputs while HDCP is active. Some restart their content from the beginning upon authentication failure. Others sources completely ignore authentication failures and transmit content anyway.

Repeaters. Adding an HDCP repeater to an installation greatly increases the odds that a setup will have problems. Repeaters can be especially prone to problems since they have responsibilities of both a transmitter and a receiver. They are susceptible to all the aforementioned inconsistencies for both sources and sinks. Add multiple inputs and outputs to the repeater and the problem gets worse. That being said, repeaters frequently get a bad rap; latent problems in the source may not come to light until it has to handle Part 2 of authentication.

HDMI. On top of all of this, HDMI has its own complexity issues. Resolution, color space, and audio problems can't be blamed on HDCP. Furthermore, HDMI problems can cause screen flashing just like HDCP.

Personally, I've seen HDMI and HDCP problems from very expensive prosumer equipment and inexpensive Asian imports alike. But there is good news. As device manufacturers have developed more HDCP-compliant devices they've gotten better at it. DCP and the Consumer Electronics Association sponsor periodic PlugFests, which are industry-wide interoperability testing events. The events have given engineers the opportunity to test their devices with those from other manufacturers. As a result, implementations have been improving. And HDCP problems that do arise are frequently solved with a [firmware](#) update from the manufacturer.

Conclusion

Implementing HDMI and HDCP can be a complex process, especially for those who underestimate that complexity going in. Fortunately for our industry and our consumers we've been continuously improving and making better products.

About the author

Robert Carter is a firmware developer for [Crestron Electronics, Inc.](#)
