An Analysis of HDCPv2 and Other Major Digital Media Copy Protection Schemes

Jeremy Barthélemy and Anurag Kamat Haldonker

Abstract—**Various copy protection schemes are explained as an overview to this report. HDCP and HDCPv2 are both compared and HDCPv2 is analyzed. A study of the strengths and weaknesses of HDCPv2 has been performed and included in the paper. We find in our report that HDCPv2 improves upon HDCP in some ways, but that there are still some potential weaknesses which will hopefully be addressed in the next specification of HDCPv2.**

Key words—**Digital media protection; HDCPv2; HDCPv1**

I. INTRODUCTION

Software differs greatly from physical items in that software can be stolen, copied, and easily distributed to thousands of people, whereas physical objects cannot. For this reason, it lies in the best interests of owners of intellectual property to have security to protect their digital media. Unfortunately, there lies a serious and persistent problem with copyright protections with things such as video games, movies, software programs, and other forms of software.

Our goal was to research various encryption technologies currently used for copyright protection. We wished to discover the weaknesses and strengths of a variety of protection schemes, particularly HDCPv2, and also to look to the future in terms of what is needed to improve the security of intellectual properties.

Most importantly, we looked at various methods of software and hardware protection schemes (like watermarking and forensic reporting) to see if and how the latest schemes implement these features. We also tried to gauge just how secure these schemes are by finding their possible vulnerabilities, their ability to adapt to these vulnerabilities, as well as their strengths.

In this paper we describe the 2nd version of interface independent adaptation of High-bandwidth Digital Content Protection (HDCP) system. The earlier version of the protocol HDCPv1 was used extensively in many devices starting around 2001 but in 2010 the master key of the protocol version was cracked which effectively made it an unsecure protocol to be used any further.

HDCPv2 overcomes several limitations and flaws of the original protocol while maintaining its practicality. Although it is being used by a number of recently introduced devices, the new version of this protocol is yet to fully take off in the market. This project will provide us with an opportunity to fully analyze the protocol and determine its advantages as well as its disadvantages over the earlier version.

II. COPY PROTECTION SYSTEMS – MAJOR PRODUCERS AND SCHEMES

Below, we analyze some well-known copy protection schemes to give some background to Digital Rights Management and encryption schemes used for the protection of intellectual property.

1. *CSS (Console CD Protection)*
   The Content Scramble System (CSS) was originally introduced in 1996 for commercial DVDs. CSS used a 40-bit stream cipher for performing encryption.
   In 1999, a group led by Jon Lech Johansen was able to create "DeCSS," a computer program which was constructed by reverse-engineering CSS.
   A major problem of CSS that was easily exploited was the key size of CSS. Being so small (40 bits), brute-force attacks were found to be particularly powerful against CSS. For example, in 2004 a 40-bit key could be found in approximately two weeks, with just a simple home computer. With botnets or hardware working in parallel, this number could be greatly reduced to even being broken in mere seconds. (Schneier 1996, p. 154.). Due to this weakness to brute-force attacks and the public release of DeCSS, CSS has been superseded by various encryption algorithms like CPRM (Content Protection for Recordable Media).

2. *AACS*
   The Advanced Access Content System was originally released in April of 2005 as the access restriction scheme for both HD DVD (56 bit key) as well as Blu-Ray (128 bit key) formats.

   AACS uses the Advanced Encryption Standard (AES) in order to encrypt content. A specific volume ID, which is based on an actual physical serial number on the disc, as well as a processing key, Km, are used as inputs, and the output of the AES encryption will be a volume unique Kvu. Kvu is then used in the decryption of encrypted title key, which is also on the disk. The title key is derived from the media key that is encoded in the Media Key Block and the volume ID. The purpose of the title key, Kt, is to perform the decryption of the actual digital content of the disc. Every licensed player contains a set of keys which are specific only to that player. If the player's keys become compromised, the publisher can simply revoke the single player's keys, which will then stop the player from being able to decrypt the protected content.

The AACS implementation was targeted by hackers in several ways. Hackers were able to attach a debugger to an AACS licensed software player and determine the location in memory of the Volume ID keys. The first device keys were extracted from a software player in 2007. After this, the processing keys for the first and second iterations of the media key block were found and posted on the Internet. AACS has been since cracked, but Blu-Ray offers BD+, which is an added protection scheme which has not been cracked yet.

3. *Cinavia*

In 2009, it was deemed a requirement for AACS to have Cinavia detection on Blu-rays. Cinavia (originally Verance Copy Management System for Audiovisual Content) is an audio watermarking system that was developed by Verance Corporation and released in 2010. The digital media protected by Cinavia have signals embedded by the owners of the intellectual property in the media, as a code that allows the player to determine if the media can be played. This is called a watermark. Watermarks are embedded in the audio as an inaudible code, which must not affect sound quality and must not be perceivable to the human ear. It must also be resistant to compression and encryption and must be able to survive a recording through a microphone (this way, if somebody illegally copies a song or movie, the watermark will still be present in the illegal copies of the media).

Cinavia is able to withstand a multitude of transformations and manipulations of the audio. These transformations can be from simple white noise as well as from hackers attempting to strip the audio signal of its watermark (in the case of having the watermark being used passively to determine the source of illegal content).

When media are played on a system with Cinavia protection, the watermark must be detected, and the device must be checked for authorization for the specific watermark. If it is unauthorized, sound may be muted, or the media may be stopped entirely.

Digital watermarks are a sort of fingerprint which are placed into intellectual property signals, which allow for the identification for the media. One main goal of the watermark is to be able to reside in the protected content and have great resistance against noise and other manipulations, while still never adversely affecting the content itself. The watermark must be easily embedded and just as easily detected by the player. Depending upon the use of the watermark, it could be desired to ensure the watermark is removed with small manipulations (not including noise), and another with only extreme manipulations that would greatly degrade the content. The first case is to ensure that if illegal content is produced, the illegal copies will not have the watermark. The second case is to allow digital content owners to pinpoint the origin of illegal copies by having different watermarks in each distribution center. Watermarks can be used as a DRM in that, if the watermark is not detected by the player, the player can willingly reduce the quality of the media. For example, if a watermark is not detected within a short time of running the video game or film, instead of playing the media in high quality, it could be played in standard quality, with no sound for illegal copies of the content.

In our research, we've considered various schemes of protection of digital media. Watermarking and Forensic Reporting are two of these schemes which caught our attention, in particular, because of their possible benefits with regard to creating players for digital media which are able to adapt to changes as opposed to simply being static and falling prey to a wide variety of attacks. Something needs to be added to digital media to improve the response to attacks and enable future protection from these past attacks, thus, some system that is able to "learn."

Watermarking is a method used sometimes in digital media generally to identify the ownership of copyrighted material. It is similar to typical steganography in that it attempts to embed some information secretly inside of some form of media, such as an image. Steganography focuses on making data imperceptible by humans whereas watermarking focuses more on maintaining the integrity against modifications of the media in efforts to remove the watermark.

Digital Watermarks are only perceptible under certain conditions. In other words, they should only be visible after a specific algorithm has been performed upon them, and they should be totally imperceptible for all other cases.

There are two main uses for digital watermarks:

<u>Source Tracking</u>

In this scenario, a specific watermark is embedded into digital media at different times of the distribution of the product. Whenever an illegal copy is found, all that is necessary is to find which watermark is present in the illegal copy and the source of the illegally copied media is detected.[14]

Detection of Modifications

If media are modified, then the watermark will disappear. The watermark can range from fragile to robust in terms of classifications with regard to the removal of the watermark when changes are made to the media. If there are very small changes, a fragile watermark will no longer be detectable using the algorithm for the watermark detection by the creators of the watermark, and so they will know that some sort of modification has occurred. Robust watermarks allow for specific classes of transformations to be detected.

4. *DPM*

There are some protection schemes that rely on the "geometry" of a CD-ROM instead of digital watermarking for protection. For example, "CD-Cops" uses DPM (Data Position Measurement) to measure the physical location of data on the disc. If a CD is stamped, it will always have the data in the same location of the disc, but burned copies have differences in the physical location of the data. DPM will detect these discrepancies in order to determine whether or not a disc was burned (and thus if it is legitimately factory-created or it was pirated).

III. OVERVIEW OF HIGH-BANDWIDTH DIGITAL CONTENT PROTECTION (HDCP) VERSION 1

The original HDCP was a scheme developed by Intel to prevent digital media from being copied as it travels connections, particularly video display interfaces, as these connections are extremely vulnerable to copying without some form of countermeasures. A master key that could be used for the production of new keys was released in September 2010, having been either reverse-engineered or leaked, although serious questions were raised as early as 2001 regarding the actual security of HDCP by Scott Crosby of Carnegie Mellon University. Scott Crosby was able to demonstrate weaknesses in the authentication protocol by explaining that if 40 KSVs (Key Selection Vectors) which are linearly independent are distributed publicly, the security of the protocol would disappear entirely.

At the ACM-CCS8 DRM Workshop, Crosby, as well as several colleagues (Goldberg, Johnson, Song), explained that the linear key exchange weakness would allow for a multitude of security threats to be enabled, including: the ability to eavesdrop on data, create new key vectors, and the ability to clone a device with just its public key.

Niels Ferguson, another researcher, claimed in the same year that it would be possible to attain an HDCP master key in just two weeks by running four computers and fifty HDCP displays in parallel.[5]

IV. HIGH-BANDWIDTH DIGITAL CONTENT PROTECTION (HDCP) VERSION 2

HDCPv2, similar to the original, has been designed to protect valuable content traversing a wire or wireless medium, from a receiver to a transmitter (for example, a Blu-ray player to a TV).

In order to perform this cryptographic protection scheme, HDCPv2 performs the following, which we will discuss in further detail below:

-Exchange and verification of public key certificates

-Establishment of shared secret keys between the transmitter side and the receiver side along with the creation of a cache of shared keys for later use in the session

-Verification that the data is not being forwarded to a remote party (locality test)

The authentication protocol phase of HDCPv2 is made up of four stages and is carried out between the HDCP transmitter and the HDCPv2 receiver. In the first stage the transmitter authenticates the receiver and initiates a key exchange (AKE). The second stage is reserved for locality check, which ensures locality on the content, where the round trip time (RTT) of the messages exchanged between transmitter and receiver should be less than 7 milliseconds. The third stage is where the session key is exchanged and the fourth and the last stage ensures authentication of the repeaters. The last step is carried out by transmitter only on HDCPv2 Repeaters where repeater brings together the topology information and sends it to the transmitter.

1. *HDCPv2 Authentication and Key Exchange*

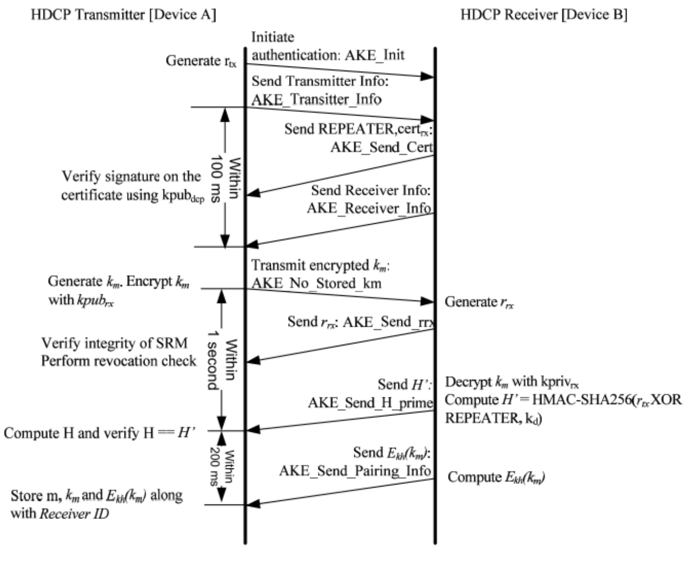

HDCP Transmitter [Device A]
HDCP Receiver [Device B]
Generate r_{tx}
Initiate authentication: AKE_Init
Send Transmitter Info: AKE_Transitter_Info
Send REPEATER,$cert_{rx}$: AKE_Send_Cert
Within 100 ms
Verify signature on the certificate using $kpub_{dcp}$
Send Receiver Info: AKE_Receiver_Info
Generate k_m. Encrypt k_m with $kpub_{rx}$
Transmit encrypted k_m: AKE_No_Stored_km
Generate r_{rx}
Send r_{rx}: AKE_Send_rrx
Within 1 second
Verify integrity of SRM
Perform revocation check
Send H': AKE_Send_H_prime
Decrypt k_m with $kpriv_{rx}$
Compute $H' = $ HMAC-SHA256(r_{tx}XOR REPEATER, k_d)
Compute H and verify H == H'
Within 200 ms
Send $E_{kh}(k_m)$: AKE_Send_Pairing_Info
Compute $E_{kh}(k_m)$
Store m, k_m and $E_{kh}(k_m)$ along with *Receiver ID*

Fig. 1. Authentication and Key Exchange (k_m not stored)[4]

The transmitter initiates the authentication process by generating a 64-bit value r_{tx} which is transmitted as an initial message AKE_Init to the receiver which is then followed by transmitter information message. The receiver responds with a request for the transmitter to send its certificate. This is accompanied with the certificate cert$_{rx}$ of the receiver; it also indicates whether it is a repeater or a receiver. Receiver then sends its information. It is to be noted that the information has to be received by the transmitter within 100 ms of sending its information otherwise transmitter will consider the receiver to be a HDCP 2.0 compliant device.

Now the authentication process can take place with or without storing the master key k$_m$. If the master key is not stored corresponding to the receiver's ID then the transmitter authenticates the receiver using the 3072-bit RSA public key kpub$_{dcp}$ present with it. If the authentication is successful then it will generate 128-bit master key, which will be encrypted to 1024-bits using receiver's public key and sent to it. In the meantime the top-level transmitter checks the integrity of System Renewability Message (SRM) using kpub$_{dcp}$ and also checks if the receiver ID has been listed in the revocation list. Failure of the integrity test or presence of revoked ID means authentication failure.

The receiver on the other side generates a 64-bit pseudo-random number r$_{rx}$ then decrypts the received k$_m$ value using its private key, executes the process of deriving the 256-bit key k$_d$ and computes a hash value H' using HMAC-SHA-256 algorithm. This H' is sent to the transmitter. The transmitter then derives the key k$_d$ and computes its own hash H which is compared with H'. If the two hash values are equal and the H' value is received within 1 second of sending the encrypted k$_m$ then authentication is said to be successful. It is also to be noted that when k$_m$ is not stored the receiver generates a key k$_h$ of 128-bits and uses it to encrypt the master key k$_m$ and sends it to the transmitter (see Fig. 2). Upon receiving this message the transmitter then saves the encrypted k$_m$ along with k$_m$ and m (which is r$_{tx}$ appended with 64 0's)

2. *Pairing Protocol*

Key pairing allows the derived key K$_m$ to be stored for later use, as asymmetric-key exchanges are time-consuming.

3. *Locality Check*

HDCPv2 allows for a localization check, a method for determining how far the transmitter is from a receiver. The purpose of this is to restrict the transmission of HDCPv2 content remotely across the Internet. For the HDCPv2.0 revision, a signal is sent to the receiver. Once this signal is sent, a timer at the transmitter starts. The round trip time from the target to the source and back must be fewer than 7 milliseconds, otherwise the timer will expire, and the session will be terminated.

The locality check follows the AKE wherein the transmitter sends a locality check value and waits for 7 milliseconds for the receiver to send a value L' which is HMAC-SHA256(r$_n$,k$_d$ XOR r$_{rx}$). The transmitter also calculates value L in the same way and compares it with L'. If L' and L are equal and if the reply is received with 7 ms then the locality check is successful.

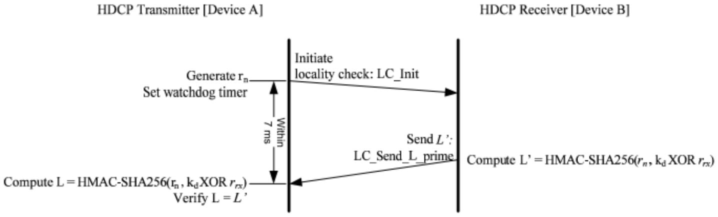

HDCP Transmitter [Device A]
HDCP Receiver [Device B]
Initiate locality check: LC_Init
Generate r_n
Set watchdog timer
Within 7 ms
Send L':
LC_Send_L_prime
Compute L' = HMAC-SHA256(r_n, k_d XOR r_{rx})
Compute L = HMAC-SHA256(r_n, k_dXOR r_{rx})
Verify L = L'

Fig. 2. Locality check[4]

4. *Session Key Exchange*

The next step is session key exchange (SKE) which has to occur only after AKE and locality check are successful. The transmitter generates and encrypts a session key k$_s$ (128-bits) to be sent to the receiver as follows

E_{dkey} (k$_s$) = [k$_s$ XOR (d$_{key2}$ XOR r$_{rx}$)]

where d$_{key2}$ is a 128-bit key derived by transmitter and r$_{rx}$ is XORed with the least significant 64-bits of d$_{key2}$

The receiver upon receiving this message derives d$_{key2}$ and decrypts the message to get the session key.

5. *Authentication of Repeaters*

Finally, the authentication with repeaters is done only when repeaters are involved in the communication. The repeater has to provide topology information to the transmitter as upstream propagation only then the transmitter sends the downstream propagation of content stream management information.

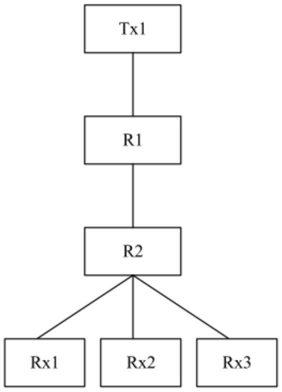


Fig. 3. Finding topology at the repeater end[4]

In the above case the repeater R1 will report depth of 2 and device count of 4 to the transmitter while the repeater R2 will report depth of 1 and device count of 3.

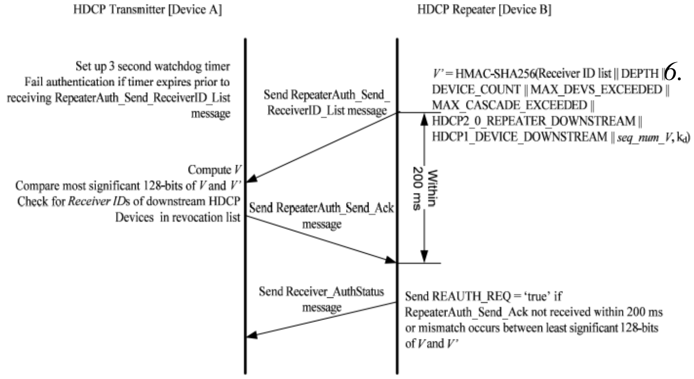

HDCP Transmitter [Device A]
HDCP Repeater [Device B]
Set up 3 second watchdog timer
Fail authentication if timer expires prior to receiving RepeaterAuth_Send_ReceiverID_List message
Send RepeaterAuth_Send_ReceiverID_List message
V' = HMAC-SHA256(Receiver ID list || DEPTH |6.
DEVICE_COUNT || MAX_DEVS_EXCEEDED ||
MAX_CASCADE_EXCEEDED ||
HDCP2_0_REPEATER_DOWNSTREAM ||
HDCP1_DEVICE_DOWNSTREAM || seq_num_V, kd)
Compute V
Compare most significant 128-bits of V and V'
Check for Receiver IDs of downstream HDCP Devices in revocation list
Send RepeaterAuth_Send_Ack message
Within 200 ms
Send Receiver_AuthStatus message
Send REAUTH_REQ = 'true' if RepeaterAuth_Send_Ack not received within 200 ms or mismatch occurs between least significant 128-bits of V and V'

Fig. 4. Authentication of repeaters[4]

A repeater has to send a hash value produced from data which comprises of receiver ID, depth and device count. This hash value V' is sent to the transmitter which compares first 128-bits of V' with the corresponding bits of the hash value V it has generated separately. If the receiver ID's are not in the revocation list then the transmitter sends the least significant 128-bits of V to the repeater. This message has to be received by the repeater within 200 ms of sending the V' value. If there is a mismatch between V and V' or if the message is not received within 200 ms, the receiver will send a message which requests reauthorization and until it is authorized its status remains as 'unauthenticated'.

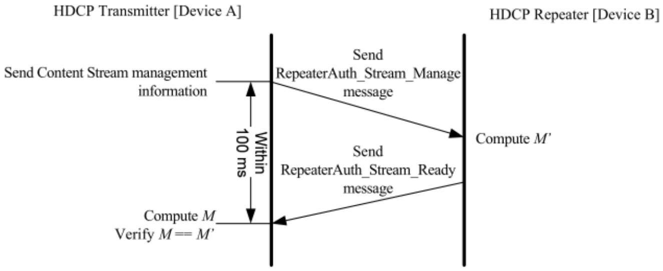

HDCP Transmitter [Device A]
HDCP Repeater [Device B]
Send Content Stream management information
Send RepeaterAuth_Stream_Manage message
Compute M'
Within 100 ms
Send RepeaterAuth_Stream_Ready message
Compute M
Verify M == M'

Fig. 5. Transmission of content management information[4]

The transmitter may send multiple content streams to the receiver all of which can be encrypted by the same session key k_s. The content stream management information has to be sent before any HDCP data is sent. Transmitter sends content management information to the receiver which calculates M' such that

M (or M') = HMAC-SHA256 (STREAMID_TYPE || seq_num_M, SHA256 (k_d))

The M' has to be received within 100 ms by the transmitter which then computes M and compare is with M'. If both values are unequal or the M' value from the receiver is not received within 100 ms then the transmitter will not send the content streams identified in the prior messages.

HDCPv2 Encryption

MPEG AV streams consist of Packetized Elementary Streams (PES). Associated PES streams are grouped in a Program. Aside from this various control, status, timing, and formatting information is transported but only the AV streams are subject to HDCP encryption.

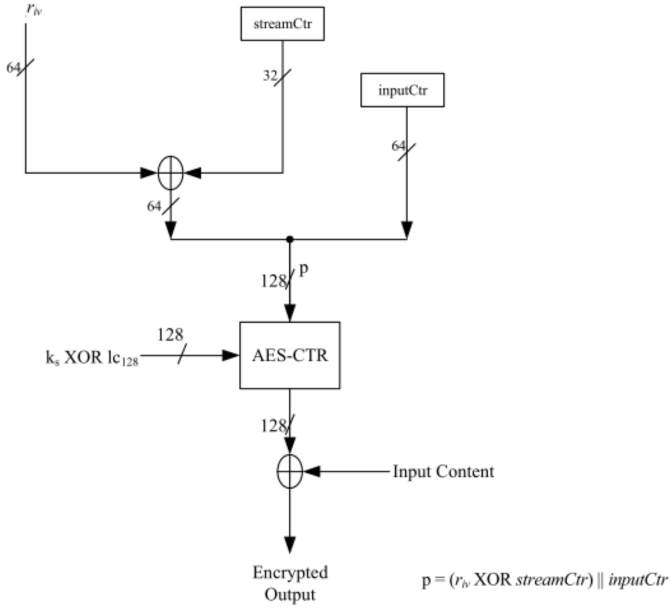
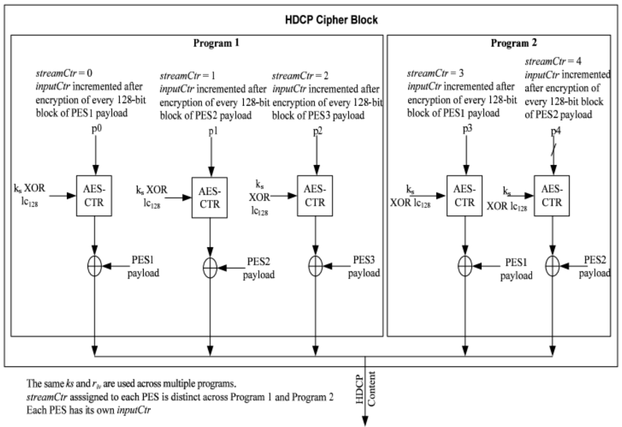


Fig. 6. HDCPv2 Cipher Structure[4]

HDCP encryption block uses AES algorithm that is operated in counter mode (AES-CTR). First the block needs to generate a 128-bit value 'p' such that
p = (r_{iv} XOR *streamCtr*) || *inputCtr*
streamCtr is a 32-bit counter which is assigned to each PES such that it is different for each elementary stream in a given program. The first stream will have *streamCtr* equal to 0 and each of the following streams will increment *streamCtr* by 1. *inputCtr* is a 64-bit counter whose value should never be reused for a given set of parameters i.e. k_s, r_{iv} and *streamCtr.*

It then will perform an XOR of the 128-bit session key k_s with the global constant lc_{128} to produce a value that can be applied to the AES-CTR algorithm as a key. The output of this algorithm is then XORed with the payload data to produced encrypted data.[3],[4]


HDCP Cipher Block
Program 1
Program 2
streamCtr = 0
inputCtr incremented after encryption of every 128-bit block of PES1 payload
streamCtr = 1
inputCtr incremented after encryption of every 128-bit block of PES2 payload
streamCtr = 2
inputCtr incremented after encryption of every 128-bit block of PES3 payload
streamCtr = 3
inputCtr incremented after encryption of every 128-bit block of PES1 payload
streamCtr = 4
inputCtr incremented after encryption of every 128-bit block of PES2 payload
p0
p1
p2
p3
p4
k_s XOR lc_{128}
k_s XOR lc_{128}
k_s XOR lc_{128}
k_s XOR lc_{128}
k_s XOR lc_{128}
AES-CTR
AES-CTR
AES-CTR
AES-CTR
AES-CTR
PES1 payload
PES2 payload
PES3 payload
PES1 payload
PES2 payload
HDCP Content

The same *ks* and r_{iv} are used across multiple programs.
streamCtr asssigned to each PES is distinct across Program 1 and Program 2
Each PES has its own *inputCtr*

Fig. 7. HDCPv2 Encryption of multiple programs[4]

V. COMPARISON OF HDCPv1 WITH HDCPv2

HDCPv2 is generally seen as an entirely different protection scheme when compared to HDCPv1. Although the two do share a few similarities, such as having the same revocation system, HDCPv2 can only be used in new technologies, and there is no plan to replace HDCPv1 in legacy systems.

HDCPv2 uses enhanced, state-of-the art cryptography (RSA, AES, HMAC-SHA256) as well as locality check to prevent proxy, it can be used in wire-line or wire-less scenarios with compressed data and global constant stored at the source and certificates stored at the sink. On the other hand its predecessor HDCPv1 used a fairly robust cryptographical procedure without any setup for locality check, worked only in wired scenarios with uncompressed data and it stored keys at the source and sink which posed a security risk.

VI. BREAKING HDCPv2: CRYPTANALYSIS

History shows us that cryptographic protocols that seem very secure can and will be broken eventually. It is extremely difficult to develop a powerful cryptographic protocol that is resistant against threats. While researching, we wondered what could be some possible vulnerabilities of HDCPv2. Below, we analyze some possible weak points that we have researched with regard to the latest specification of HDCP, based upon the latest specification.

1. Possible Problems with the AKE

In short, this section of the protocol verifies that the HDCP receiver is via a public key certificate, and a master key is exchanged.

There is no authentication for the transmitter. Although this does not seem to allow for any malicious attacks, we believe that it could be possible for hackers to build their own transmitters in order to gain insights into further possible weaknesses in the protocol, which are not so easily discerned. For example, the session key K_d is based solely upon inputs from the transmitter, so it could be possible that this is overlooked, as it does not seem to be a problem now, only for hackers to take advantage of this in the future. Our solution would be to add authentication for the transmitter. Although this would lead to an added overhead in the initial AKE phase, and thus slower transmission initialization, we feel that developing a simple means of verifying that a transmitter is valid and authorized to transmit HDCP content is the best solution, and that the benefits outweigh the costs.

2. Possible problems with the Locality Check

One weakness is that we are limited to 7 ms for a round-trip communication between the transmitter and the receiver. For slower devices, this could prove to be a problem. But if the period is too long, then valuable information could be illegally piped to different devices located far away geographically. HDCP, however, does provide a means of protecting against this scenario by allowing for a second option, which will perform the locality check method that allows the receiver to compute L' at its own leisure (and send RTT_Ready when it is ready for the least significant 128 bits of L), the receiver can simply respond quickly with the most significant 128 bits of L'.

Because of this feature, it seems as though the locality check is acceptable, although it is not certain that there are not some small possible approaches of attacking the protocol for locality check.

There lies an interesting situation, regardless, of whether or not it is possible to trick the transmitter and the receiver into using the opposite protocol option for locality checks.

Perhaps it could be possible for an attacker to convince the transmitter to run the protocol for slower devices, and the receiver to run the simplified one; the message flags are never once authenticated during the AKE phase and to modify the message flags, the transmitter will send R_n, and allow us as much time as necessary to perform the computation. Then, the value R_n could be passed on to the receiver to wait for the receiver's calculations for L'. This may take as long as necessary, because the transmitter is still waiting for the RTT_Ready message. Once ready, we can send the RTT_Ready message, and the transmitter will send the least 128 significant bits of L, and we must reply within 7 ms of receiving this message with the most significant 128 bits which the receiver in a distant geographic location has calculated.

Then, we could simply pass the intellectual property such as a movie on to wherever in the world the receiver is. Of course, this scenario does assume that we are able to trick the transmitter and the receiver into operating in opposite modes, but it is a possible avenue of attack that needs to be addressed in order to assure the security of the protocol.

This is a difficult issue to solve, if in fact this possible weakness is exploited. The only way to totally eradicate this threat would be to shift to the original option of having the 7 ms check locality response to the transmitter required and to remove the second option.

Preferably, this would be avoided so as to not alienate users of HDCPv2 with slower devices, although it is always a possibility if the locality check is exploited ubiquitously. [7]

VII. CONCLUSIONS

After an analysis of copy protection schemes, we find that HDCPv2 is vastly superior to the earlier version.

Among the major differences of these two algorithms, the new authentication and key exchange phase as well as the locality check phase are the two which appear to be most significant, along with superior encryption.

Much like the predecessor of HDCPv2, however, we find that it is still quite likely for hackers to eventually discover a weakness and bypass the copy protection provided by it, unfortunately. Although the algorithm may look very strong, "strong" algorithms in the past have also been broken.

VIII. GLOSSARY

Cert$_{rx}$: Certificate issued to the receiver by DCP LLC which has 1024-bit RSA public & private keys

Ctr$_i$: counter value for iteration i

dkey$_i$: key derived by transmitter depending on Ctr$_i$ (128-bits)

H : Hash value at transmitter side (256-bits)

H': Hash value at receiver side (256-bits)

k$_d$: Derived key (K_d = dkey$_0$||dkey$_1$) (256-bits)

k$_h$: Receiver generated key for encrypting k_m (128-bits)

k$_m$: Master key (128-bits)

kpriv$_{rx}$: Receivers secret RSA private key (1024-bits)

kpub$_{dcp}$: RSA public key for authenticating receiver when master key is not stored (3072-bits)

k$_s$: Session key (128-bits)

L : Hash value at transmitter side for locality check (L = HMAC-SHA256(R_n,k_d XOR r_{rx}) (256-bits)

L': Hash value at receiver side for locality check (L = HMAC-SHA256(R_n,k_d XOR r_{rx}) (256-bits)

r$_n$: Transmitter generated pseudo-random number for locality check (64-bits)

r$_{rx}$: Receiver generated pseudo-random number for key exchange (64-bits)

r$_{tx}$: Transmitter generated pseudo-random number for key exchange (64-bits)

V: Hash value at transmitter side for authenticating repeaters

V' : Hash value at receiver side for authenticating repeaters

inputCtr: Input counter (64-bits)

lc$_{128}$: Secret global constant (128-bits)

r$_{iv}$: Initialization vector (64-bits)

streamCtr: Counter value (32-bits)

IX. REFERENCES

[1] CD/DVD/Media-Protections http://www.cdmediaworld.com/hardware/cdrom/cd_protections.shtml

[2] Copyright Protection Scheme for Digital Television Content International Journal of Information Technology, Vol. 11 No. 9 2005

[3] Digital Content Protection LLC., High-Bandwidth Digital Content Protection - Content Protection for Next Generation Scenarios

[4] Digital Content Protection LLC., High-bandwidth Digital Content Protection System Specification, Revision 2.2, July 18, 2011,

[5] Digital Content Proection LLC., High Bandwidth Digital Content Protection System, Revision 1.4, July 8, 2009.

[6] Gigaom., DRM FAIL: Five Broken Copy Protection Schemes, September 2010.

[7] Green, Mathew. A Cryptanalysis of HDCPv2.1, August 27, 2012,

[8] Halderman, Alex, et. al, Lessons from the Sony CD DRM Episode,

[9] Hassanien, Abou Ella., A Copyright Protection using Watermarking Algorithm, INFORMATICA, 2006, Vol. 17, No. 2, 187-198, 2006 Institute of Mathematics and Informatics, Vilnius.

[10] Hindawi, Robust Signature-Based Copyright Protection Scheme Using the Most Significant Gray-Scale Bits of the Image
International Journal of Information Technology, Vol. 11 No. 9 2005

[11] Hwang, Ren-Junn., A Digital Image Copyright Protection Scheme Based on Visual Cryptography, Tamkang Journal of Science and Engineering, Vol. 3, No. 2, pp. 97-106 (2000)

[12] Kocher, Paul, et. al., Self-Protecting Digital Content Cryptography Research, Inc., 2002-2003.

[13] Lambert, Eric., Vintage Computing, Old-School PC Copy Protection Schemes, August 8th, 2006.

[14] Verance, Verance Copy Management System: Presentation to CPTWG ARDG, 10 April 2003.

[15] Ayera, Michael.,AACS Overview Presentation to CPTWG , Advanced Access ContentSystem (AACS), July 22, 2009