Andreas Koefoed-Hansen
20062154
u062154@cs.au.dk
Tim Rasmussen
20061947
tiras@cs.au.dk

Advanced Access Content System - Secure or not?

December 18, 2009

Contents

1 Introduction

This rapport shed light on the subject Advanced Access Content System (AACS), with a special focus on the security in the system. We will cover some of the main mechanisms that helps making this system secure (or insecure), starting with a short introduction to the subject. The whole idea with the introduction is to give an idea of what the AACS is all about and why it was created before we begin explaining the details in the security. After the introduction we will list and explain the different algorithms and keys used later in the rapport. This section can later be used as a look-up table, if you cannot remember the definition of an algorithm or key used. We will continue describing different security mechanisms such as Media-Key-Block, Subset-Different Trees, Traitor Tracing and the authentication process. At last we will describe some of the attacks made on the system, and how the attackers succeeded. Based on these attacks, we will shortly describe who they tried to make Blue-ray discs more secure by adding an extra layer, and how this extra layer once again was broken.

We will end this rapport with a conclusion containing an evaluation on the security of the system, and why it was broken.

The rapport is mainly based on the two specification Intel et al. (2009a) and Intel et al. (2009b) from the AACS LA, so when no sources are added, these are the ones refered to.

2 Introductory description of AACS

Video media has changed a lot over the years. In the 1970 VHS was one of the media which contains video, and later on in the 1990th it became the preferred video tape format[1]. At the time there were no copy protection, so if you had to VCR's you could easily copy one tape to another. Around 1996 encrypted DVD's started to show. They were protected by Content Scramble System (CSS), which is a Digital Rights Management (DRM) scheme used on almost all commercially produced DVD-Video discs. With the introduction of HD-DVD and Blu-ray Discs, the CSS algorithm has been replaced with a new DRM scheme AACS, which is based on AES encryption.

AACS is developed by AACS Licensing Administrator (AACS LA), a consortium that includes Disney, Intel, Microsoft, Panasonic, Warner Bros., IBM, Toshiba and Sony[2]. The reason why AACS LA chose to make a new DRM scheme instead of just continuing using CSS was because they wanted to improve the security of the media. In 1999, an addressing flaw has led to at total bypassing of the CSS, and it therefore became easy to copy encrypted content

[1]Wikipedia (2009d, VHS)
[2]Wikipedia (2009b, Advanced Access Content System)

from one disc to another.

	Content Scramble System	Advanced Access Content System
Algorithm	Proprietary	AES
Key strength	40-bit	128-bit
Certificate revocation	No	yes

Table 1: Comparison of CCS and AACS. Source Wikipedia (2009c, Security of Advanced Access Content System)

By comparing AACS with CSS, you see that key strength in AACS is much higher, and therefore should be much more secure against attack[3]. Another difference between AACS and CSS is the way the devices decryptions keys are organized. Using CSS, all players of a given model are provided the same decryption key. Content on the disc is encrypted under a title-specific key, which again is encrypted under each model keys. This means that the disc contains a lot of encryption keys that is being used to make the connection between the device and the title-specific key. In comparison, AACS provides each individual player with its own decryption key called a Device Key. This makes it possible to revoke a given player, and thereby make it impossible for the player to decrypt new Title Keys. It is also possible to revoke players using CSS, however here not only a specific player is revoked, but all players from a particular model are revoked from decrypting the Title Key, and thereby many users lose the capability to playback. AACS also gives the ability of Traitor Tracing, which means that if an attacker is publishing a decrypted Title Key or a copy decrypted with the Title Key, then it is possible to track down the Device Key from the Title Key. This is possible because the standard allows short sections of the movie to be encrypted with different keys (also called sequence keys). A certain player will only be able to decrypt one version of each section, and thereby leaving a watermark in the Title Keys he is publishing. This watermark can be used to track down the player and the revoke it.

2.1 Content Encryption

We have now explained some differences between CSS and AACS and now turn our attention towards explaining the basic idea behind AACS. We will start by explaining the encryption process and afterwards turn our attention towards the decryption process.

The encryption process starts when the owner of the content to be protected, provides the Licensed Replicator with one or more titles and associated Usage Rule. The Idea is now that the Licensed Replicator selects a secret random Title

[3]Wikipedia (2009c, Security of Advanced Access Content System)

Key for each title to be protected, and uses this key to encrypt the content of the corresponding Title. The Licensed Replicators decides whether to use the same Title Key on all instances of pre-recorded media to encrypt the Title or use different Title Keys for each instance.

It is also the job of the Licensed Replicator to assign an unpredictable identifier (Volume ID) to the protected title and include it on the pre-recorded medium. It is important that the Volume ID is stored on the pre-recorded media in such a way that it cannot be duplicated by consumer recorders. We will return to the Volume ID later in the section. It is the job of the AACS LA to provide the Licensed Replicator with a Media Key Block (MKB), a secret Media Key, a Sequence Key Block (SKB) and a corresponding set of secret Media Key Variants for each title. It is very important that the Licensed Replicator has the newest version of the MKB, to make sure that newly compromised devices keys cannot be used. The Licensed Replicator now has all information need to do the actual content encryption. This is done by calculating a cryptographic hash of the Media Key and/or the Media Key Variants and the Volume ID, and using it to encrypt the titles Title Key. The encrypted content, encrypted Title Keys, signed Usage Rules and MBK are all stored on the pre-recorded medium.

2.2 Content decryption

To control the access of the digital media, AACS uses Advanced Encryption Standard (AES) to encrypt content under one or more Title Keys. The Title Keys are stored on the pre-recorded disk, and can by licensed devices be derived. The devices derive the Title Keys by first deriving the Media Key, which is encoded in a Media Key Block (MKB). To do this, the system relies on a subset different tree, which is basically a lot of keys arranged in a tree that is used to find all other keys except its parent key. It is also the use of the subset difference tree that makes it possible to revoke a given Device Key. This is simple done by encrypting the MKB with the Device Keys parent key. When the Media Key is derived from the MKB, the Media Key is used together with the Volume ID to get the Volume unique key ($K_v u$). The Volume Unique key is then used to decrypt the Title Key, which can then be used to decrypt the encrypted content on the disc. The idea is showed in figure 1.

2.3 Security Issues

Even before its use, security researchers have doubted AACS ability to withstand attacks, and they were right[4]. Since the release of the AACS specification, different attacks have been performed. Most of the attacks are based on memory snooping performed on a PC, where hackers are looking at the memory to find Title Keys, Processing Keys and even Device Keys and Host Private Key[5]. As it seems, the only two solutions to this problem is either by disallowing PC to

[4]Wikipedia (2009c, Security of Advanced Access Content System)
[5]Wikipedia (2009a, AACS encryption key controversy)

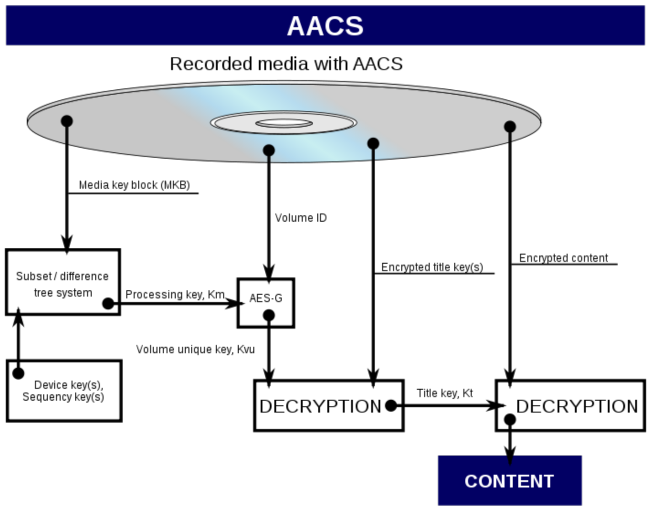


Figure 1: AACS decryption process. Source Wikipedia (2009b, Advanced Access Content System)

playback the media or by using trusted computing base (TCB). We will return to this subject in the section about attacks on AACS.

3 Definitions

In this section we will describe the different cryptographic components AACS is build on. This will include encryption schemes, hash functions and digital signatures. We will also cover the terminology used in AACS which will include a short description of keys and ids.

These definitions will be used in the later sections of this report and has an important role in understanding AACS.

3.1 Cryptographic Functions used in AACS

There are various protection mechanisms in AACS most of which builds on already known encryption or hashing functions, such as AES (Advanced Encryption Scheme) or SHA-1.

AES:
In the following section the k will represent a 128-bit key and d will be some data that will either be encrypted or decrypted based on the type of the function. When AES is used for managing cryptographic keys, it will be used in ECB

(Electronic Cookbook) mode. The encryption algorithm using ECB is represented by the function: $AES-128E(k,d)$. The function returns the encrypted result which has a length of 128 bit. The decryption algorithm is somewhat similar, it is represented by the function: $AES-128D(k,d)$. The function returns the 128-bit decrypted result. The data that get send to the functions will in both cases have to be of length 128 bit.

When AES is used for encryption or decryption the AACS content, it is used in Cipher Block Channing (CBC) mode. This means that the AES will be used as a stream-cipher and the data that will be encrypted or decrypted does not have a bound size. The encryption algorithm for encryption using AES in CBC mode will be represented as the function: $AES-128CBCE(k,d)$, where d is a frame of data. The function returns the encrypted frame. The decryption algorithm is represented by the function: $AES-128CBCD(k,d)$, d is the encrypted data frame, and the result of the function will be the decrypted frame.

AES is also used as a one-way function: $AES-G(x_1,x_2)$, where x_1 and x_2 is 128-bit values and the function returns a 128-bit result. The function uses AES-128D on the two inputs where x_2 will act as the key and x_1 as the data. In the end x_2 will be XOR´ed with the result of the $AES-128D$ and returned. This can be written as $AES-G(x_1,x_2) = AES-128D(x_1,x_2) \oplus x_2$. The figure below shows how the function works:

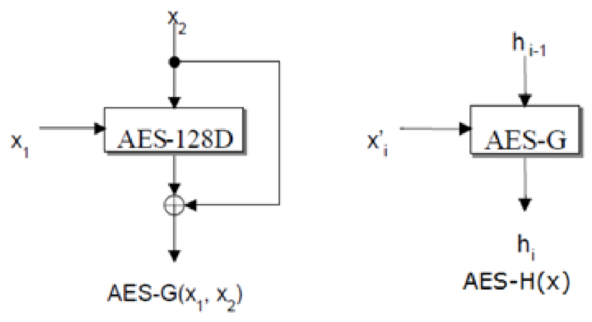


Figure 2: One-way function AES-G and hashing function AES-H. Source Intel et al. (2009a, Introduction and Common Cryptographic Elements Book)

The one-way function is primary used for hashing in AACS, this is done by the function $AES-H$ that is pictured above. Before hashing the data is partitioned into 128-bit blocks. If there is a remainder after the data has been partitioned, the last part is padded so that it will get a length of 128-bit. The padding used is the same as in the *SHA-1* method, where the first bit will be 1, followed by 0's. The function for hashing using AES is described as: $AES-H(x) = h_n$, where h_n is the result of the hashing function. The individual blocks is hashed by: $h_i = AES-G(x'_i, h_{i-1})$.

For purposes of generating a cipher-based message authentication code (CMAC), used with e.g. protection of integrity of information, the function $M = CMAC(k, D)$ is used. D is the data that needs to be authenticated and M is the resulting MAC. For computation of the MAC the need of a symmetric key block cipher is needed in AACS, AES is used for that purpose.

Digital Signatures:
In this section we will describe the functions used in AACS used with respect to digital signatures. This is primarily used with certificates and signing of content. The digital signature scheme is based on elliptic curves where the public key K_{pub} is a point of the curve and the private key K_{priv} is a scalar value that satisfies that: $K_{pub} = K_{priv} * G$. The function for creating a digital signature is represented as follows: $S = AACS\_Sign(K_{priv}, D)$ where D is the data that will be signed and S is the resulting signature. For verification of a signature the function: $AACS\_Verify(K_{pub}, S, D)$ is used. D is the data and S is the signature. All devices has the AACS LA's root public key, denoted $AACS\_LA_{pub}$.

3.2 Terminology used in AACS

In the specification of AACS there are a lot of different keys some of them are derived from other keys. To reduce the confusion we will make a lot of these and give a short description to each of them. Figure 3 gives an overview of the relation between the different terms.

Bus Key (K_b)**:** Is used by the host to verify the integrity of the values and keys sent from the Drive, and to protect the data key. The key is calculated as a part of the authentication process between the host and the drive.

Data Key: A key used to encrypt and decrypt AACS-protected sectors while in transmission to and from the Drive.

Media Key (K_m) / **Media Key Variant** (K_{mv})**:** A key that is used to unlock the Title Keys stored on a media that contains titles protected by AACS. The Media Key is computed by successfully processing a MKB. The Media Key Variant is computed by using the Sequence Key Block (SKB) and can be used instead of the Media Key.

Media Key Block (MKB): A critical component of the subset difference tree key management system. The MKB is a data block that provides access to a common key (Media Key). It is accessible by any device that contains the necessary secret keys and has not been revoked.

Subset Difference Tree: A tree-based key management system based on broadcast encryption.

Title Key (k_t)**:** The key used to encrypt and decrypt a title. When decrypting, the Title key gives access to the content.

Processing Key: This key is used to decrypt the Media Key. It is stored in the subset difference tree in the MKB block and can be accessed, if the Device Key used to get it, is not revoked.

Volume Unique Key (VUK/K_{vu}) / **Volume Unique Key Variant** (K_{mv})**:** The key is calculated by using the *AES-G* with inputs of Media Key (K_m) and the Volume ID (VID). It is used for decrypting the Title Keys (K_t). The Volume Unique Variant is calculated using the Sequence Key Block (SKB) and can be used instead of the Volume Unique Key.

Volume ID (Vid) The Volume IDs are unique identifiers that are stored on pre-recorded discs. The idea behind the use of the Volume ID in the decryption process is to prevent simple bit-by-bit copying. This can be done, because the Volume ID is required to complete the decryption process of the encrypted content. Without the Volume ID, the device will not be able to play the disc. For the device to read the Volume ID, is has to contain a cryptographic certificate, which is a Host Private Key signed be the AACS LA. The Volume ID cannot be written to a media using an ordinary recorder available to consumers, but only on the recorders used by the licensed replicators. Hackers have claimed that by modifying the firmware of a HD DVD reader, it is possible to circumvent the Volume ID and thereby decrypt the Title Key without the Volume ID. We will return to this subject in the section about attacks on AACS.

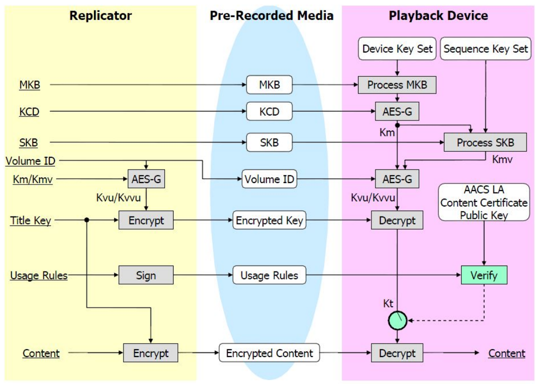


Figure 3: Shows how the encryption and decryption are handled. Source Intel et al. (2009b, Pre-recorded Video Book)

4 Security in AACS

We will in this section describe some of the important security mechanisms in AACS. We start be give a more detailed description of the decryption process. Afterwards we turn our attention towards the Media Key Block, Subset-Difference Tree and the whole process of getting the Media Key. Then we will describe the authentication process between the drive (hardware) and the host (software), and how this authentication is use to derive the Volume ID from the disc. We end this section by describing the Sequence Key mechanism and how this can be used in Traitor Tracing.

4.1 Content Decryption

Each licensed manufacturer receives 253 secret Device Keys, denoted $K_{d\_0,...,K_{d\_n-1}}$, that will be included into the player. The player reads the MKB (Master key block) from the medium and uses the Device Keys to process the MKB and calculate the Media Key. All Device Keys should be treated as highly confidential.

The media can contain multiple titles which are encrypted with the Title Keys. The player gets the Title Keys by calculating the cryptographic hash of the Media Key and the Volume ID from the media. The Title Keys are used to decrypt the titles and the content can be accessed by the player.

4.2 Media Key Block (MKB)

The Media Key Block is generated by AACS LA and allows all players to used their Device Keys to calculate the Media Key (K_m). If a Device Key is compromised, the MKB can be updated so that a player using the key is unable to calculate the correct K_m. This action is called ‘revoking‘ and ensures the integrity of the AACS system.

4.3 Subset-Difference Tree

The MKB is using a Subset-Difference Tree that contains large amounts of keys. The system has a master tree of keys where each Device Keys contains exactly one unique ‘Leaf Key‘. Given a set of Device Keys the device can derive every key in the master key except the ones on the path from the leaf and the root. This means that a device cannot derive its own Leaf Key but any other device in the tree can.

For each sub-tree in the master tree there is another system of keys. This means that in the level just below the root there are two sub-trees that each has their own tree of keys. The height of the adjacent trees is the same as their related sub-trees. The extra set of keys can be viewed as more levels on top of the master tree. In the figure below it is illustrated how the layers are arranged.

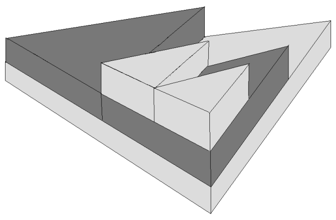


Figure 4: Levels in the Subset-Difference Tree

The concept of having more trees is introduced because it gives a very powerful toolset for revoking Device Keys.

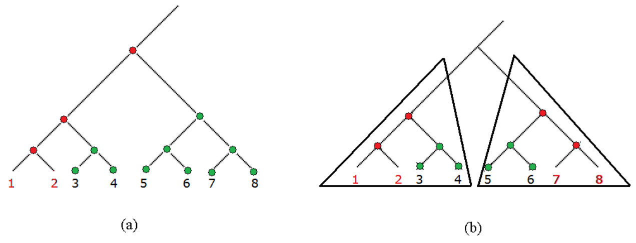


Figure 5: Illustration of subsettrees

Let us consider the case where we only have one tree and want to revoke Device Keys. Figure 5a shows the branch of the tree with Device Keys sets and the leaf nodes labeled in the bottom illustrates different devices. We want to revoke the devices at position 1 and 2. This is done by encrypting the processing key with the key that is parent to both 1 and 2, which neither of the two can decrypt. This method only works if the devices to be revokes are siblings in the same sub-tree.

Let us look at another scenario; we want to revoke both 1 and 2, and 7 and 8. This is done by using the sub-trees that was described before. Figure 5b shows how the branch now is split into two sub-trees, allowing us to apply the method from before to each of the trees. Now the processing key in the left sub-tree is encrypted using the key above device 1 and 2, and the processing key in the right sub-tree is encrypted with the key above device 7 and 8.

The name of this method comes from the idea of splitting the key-space into sub-trees (subsets), where each subset has a set of contiguous revoked nodes, which is called the subset-difference. The idea for the subset-difference system comes from the paper by Dalit Naor, Moni Naor and Jeff Lotspiech[6]. The paper describes both subset-difference and traitor tracking and proves the security of these methods.

For each subset in the MKB there is a corresponding 16-bit key in the key data part of MKB. A subset difference is encoded using two masks u and v denoted m_v and m_u. u and v are nodes in the tree and the masks denotes the path to the nodes from the root by interpreting the mask as a guide where '0' means go down the left branch and '1' means go down the right branch.

A subset-difference applies to a device if the u node is on the path from the device node to the root, but v node is not.

The Media Key K_m is calculated using the following function:

$$K_m = AES - 128D(K, C) \oplus (000000000000000000000000_16||uv) \qquad (1)$$

K is the Processing Key and C is the 16 bytes of key data stored in the key data part.

4.4 Authentication between Drive and Host

Since most attacks on AACS are performed on a PC, we will use some time, explaining how a Drive (AACS optical drive) and PC Host authenticate each other, and thereby makes a connection. It is important to understand that in a PC system the Licenses Drive and the PC Host works together as the recording device and/or playback device for AACS content. In the rest of the section we will refer to the PC Host as the Host. After describing the authentication process, we will turn our attention towards how to get the Volume ID, and how Bus Encryption is performed.

There are some constrains that the Drive and the Host should respect in order to perform an AACS Drive Authentication. A Licensed Drive must contain the $AACS\_LA_{pub}$, a Drive Certificate and a Drive Private Key, while a Host must contain the $AACS\_LA_{pub}$, a Host Certificate and a Host Private key. Since the Drive and Host certificate looks like each other, we will only explain the content of the Drive certificate in detains, and only describe where the Host certificate differs from the Drive certificates. This section is bases on the AACS specification[7].

4.4.1 Drive Certificate

The Drive Certificate is a collection of different important data, like *Type, BEC, Length, Drive ID, Drive Public Key, Signature Data(*$Drive\_Cert_{sig}$*).* The

[6]Naor et al. (2001)
[7]Intel et al. (2009a, Chap. 4)

Type is used to indicate a first-generation Licensed Drive, while the BEC is only a single bit to indicate whether the Licensed Drive is capable of performing Bus Encryption. Length contains information about the length of the certificate data including signature while Drive ID is a unique identifier. The Signature Data($Drive\_Cert_{sig}$) is the certificates signature. It is verified by using the $AACS\_LA_{pub}$ and the $AACS\_Verify()$ function. The $AACS\_Verify()$ uses the $AACS\_LA_{pub}$, $Drive\_Cert_{sig}$ and $Drive\_Cert$ to verify the correctness of the signature.

4.4.2 Host Certificate

The only difference between the Host certificate and the Drive certificate is that the host certificate contains a Data Key Settable (DKS) bit. The DKS bit is a flag used to indicate whether the Host is permitted to establish a specific Data Key.

4.4.3 Authentication algorithm (AACS-Auth)

The idea behind the AACS-Auth algorithm is that the Licensed Drive and the Host verifies that each is an AACS compliant device, that is capable of signing and verifying digital signatures and has a valid certificate signed by the AACS LA. Apart from making sure that both the Licensed Drive and the Host has a valid certificate signed by the AACS LA, it also verifies that each counterpart has not been revoked. This is done by looking at the Drive Revocation List (DRL) and Host Revocation List (HRL), which is a part of the Media Key Block. The Licensed Drive performs the verification by first making sure that is has the newest version of HRL, if this is not the case; it stores the HRL from the MKB. When it has the newest version of the HRL, It makes sure that the Host is not listed in the HRL. If it is, the authentication fails. The same verification is done by the Host, but just by using the DRL.

If the Licensed Drive is capable of Bus Encryption, it is its job to verify that the host is also capable of Bus Encryption. We will return to Bus Encryption later in this section.

At the last step of the authentication process, both the Licensed Drive and the Host shares the same information, which they use to calculate the Bus Key. So when the authentication is successful, both the Licensed Drive and the host haves a Bus Key (BK), and is now ready to continue exchanging further data. Figure 6 show the whole process.

4.4.4 Getting the Volume Identifier

The Volume identifier is read by the Licensed Drive, and securely transferred to the Host. To do this, the Licensed Drive and the Host has to perform a successful authentication. When this is done, the Host Request the Volume ID.

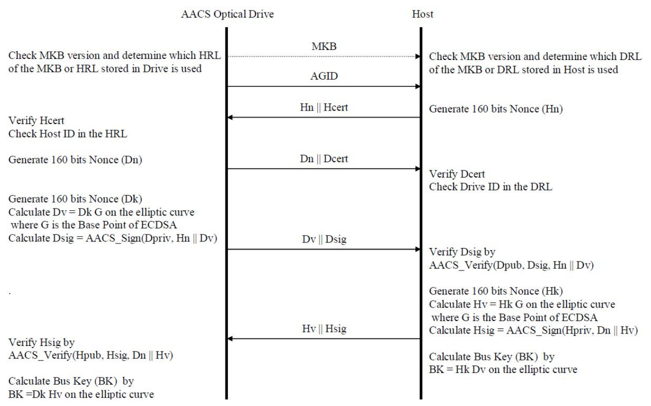

AACS Optical Drive
Host
MKB
Check MKB version and determine which HRL of the MKB or HRL stored in Drive is used
Check MKB version and determine which DRL of the MKB or DRL stored in Host is used
AGID
Hn || Hcert
Generate 160 bits Nonce (Hn)
Verify Hcert
Check Host ID in the HRL
Generate 160 bits Nonce (Dn)
Dn || Dcert
Verify Dcert
Check Drive ID in the DRL
Generate 160 bits Nonce (Dk)
Calculate Dv = Dk G on the elliptic curve
where G is the Base Point of ECDSA
Calculate Dsig = AACS_Sign(Dpriv, Hn || Dv)
Dv || Dsig
Verify Dsig by
AACS_Verify(Dpub, Dsig, Hn || Dv)
Generate 160 bits Nonce (Hk)
Calculate Hv = Hk G on the elliptic curve
where G is the Base Point of ECDSA
Calculate Hsig = AACS_Sign(Hpriv, Dn || Hv)
Hv || Hsig
Verify Hsig by
AACS_Verify(Hpub, Hsig, Dn || Hv)
Calculate Bus Key (BK) by
BK = Hk Dv on the elliptic curve
Calculate Bus Key (BK) by
BK =Dk Hv on the elliptic curve

Figure 6: AACS Authentication algorithm. Source Intel et al. (2009a, Introduction and Common Cryptographic Elements Book)

When receiving the request, the Licensed Drive reads the Volume ID from the media, and calculate a MAC using the $CMAC()$ function. The $MAC\ Dm$ is calculated by:

$$Dm = CMAC(BK, VolumeID) \tag{2}$$

where BK is the Bus Key. When Dm is calculated, the Licensed Drive sends the Volume ID and the Dm to the Host, who calculate Hm the same way as Dm, and verifies the $Dm == Hm$. If the is the case, the Host may trust the Volume Id, otherwise the Host must stop the processing of the media.

4.4.5 Bus Encryption

If both the Licensed Drive and the Host I capable of Bus encryption, an extra layer of encryption is added, and it is the job of the Licensed Drive to make sure that protected sectors are further encrypted on-the-fly. Protected sectors that requires Bus Encryption are easy to find, since it sectors contains a flag in the sector header that determines this. This also means that But Encryption is not applied to an entire AACS-protected sector, but only those requiring it.

The Bus Encryption do not use the Bus Key calculated during the authentication, but instead data is encrypted with two 128-bit Data Keys, where one is for reading and the other is for writing. Bus Encryption is performed using $AES-128CBCE$ and decryptions by using $AES-128CBCD$, both are Cipher Block Chain's (CBC) with an initial vector iv_0.

Since we only look at pre-recorded discs, we will only explain the calculation of the Read Data Key (K_{rd}). The Data Read Key is calculated from the Drive Seed S_d and the Media ID, ID_m, but since pre-recorded media do not contain a

Media ID, we use the Volume ID instead. This means that the Data Read Key is calculated by:

$$K_{rd} = AES - 128E(S_d, ID_m) \tag{3}$$

where the Licensed Drive computes the Drive Seed by:

$$S_d = AES - G([AACS_Drive_{priv}]_{msb\_128}, C_{mfg}) \tag{4}$$

C_{mfg} is a confidential 128-bit constant, which is picked by the manufacturer and places in the Licensed Drive.

4.5 Sequence Key and Traitor Tracing

Sequence Keys is another protection mechanism in AACS, but not as we know it from Device Keys and the Media Key Block. Instead it helps the AACS LA to revoke the right devices, when an attacker publishes a decrypted Title Key or content on the Internet. This can be done since the standard allows different version of short section of the movie. These sections are encrypted uses different key and when a device is trying to decrypt the movie, it will get a special sequence key which is later used to decrypt (not the Sequence Key it selves, but the Sequence Key is used to calculate the Volume Varian Unique Key (K_{vvu})) a special version of the movie section. This means that the attacker leaves a mark in the Title Key or content, which can be used to track down the Device Key used and revoke it. This process is called "Traitor Tracing" and we present a more detailed description later in this section.

4.5.1 Introduction to Sequence Key

This section is based on the AACS specification[8]. It is the job of AACS LA to provide the manufactures of the devices with a set of Sequence Keys. The set stored on the device is chosen from a large $65536x256$ matrix, where each cell is a different Sequence Key. A single device has one key in each column, which means that a device contains 256 Sequence Keys. The principle in this way of distribution the keys, is that no two devices has many keys in common. This means that even if the system has been heavily attacked, and many of the keys have been revoked, innocent devices will still be able to decrypt the Title Key.

The Sequence Key Block (SKB) on a pre-recorded media is signed by AACS LA, and allows devices with a compliant set of Sequence Keys and Media Key to calculate the Variant Data D_v. The Variant Data is later used to calculate the Media Key Variant (K_{mv}). The SKB is divided into columns, which makes it possible distinguish between legal keys and revoked keys.

The way the Variant Data is found, is by start looking at the first column, which contains an encrypted Variant Key in every uncompromised Sequence

[8]Intel et al. (2009b, Chap. 4)

Keys cell. If the devices key have not been compromised it can decrypt the Variant key, otherwise it will decrypt a key called a Link Key that points to another column in the SKB. To continue to the next column the device needs both the Link Key and its Sequence Key in that column. See Figure 7. The Sequence Key used in the process of getting the Variant Key is not the Sequence key stored on the devises, but a Sequence Key calculated from the Sequence Key matching the column and the Media Key. The device Sequence Keys are numbered $K_{s\_i}(I = 0, 1, \ldots, 255)$, which means that the Sequence Key for column i is calculated by:

$$K_{ms\_i} = AES - G(K_m, K_{s\_i}||0302153EE3EC7524_{16}) \tag{5}$$

Where K_{ms_i} is the Media Sequence Key which is the used in the decryption. The process of deriving the Variant Key continues the same way in each column, and if the device has not derived some number of columns (depends on the actual number of compromised keys) the AACS LA know that only compromised devices would reach this link. At this time all innocent devices will have found and decrypted the Variant Key while compromised devices have decrypted 0. The device knows that if it decrypts 0 it cannot continue the process of calculating Media Key Variant (K_{mv}) and therefore it stops, otherwise it will calculate the Media Key Variant by

$$K_{mv} = AES - G(K_m, D_v||041826fa7749_{16}) \tag{6}$$

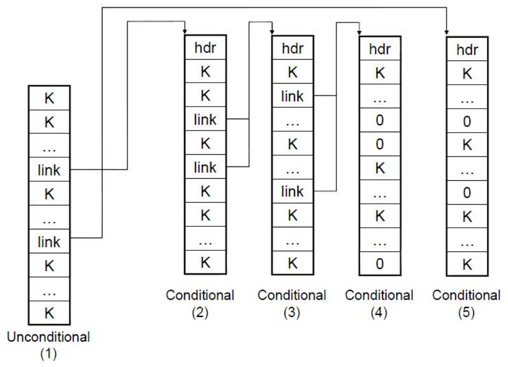


Figure 7: Example of Sequence Key Block. Source Intel et al. (2009b, Prerecorded Video Book)

4.5.2 Traitor Tracing

As described above, this technique can be use to Traitor Tracing. Sequence keys were first added to the specification later, so the first Traitor Tracing mechanism was a bit different. To understand how this Traitor Tracing system works, it is important first to understand the attacker. An attacker can publish his Device Key, but with great risk that the AACS LA will find and revoke it[9]. Therefore attackers try to keep their Device Keys secret. This is done by making a web site, where people can upload header information from a disc they want decrypted. The attacker then uses his device to extract the Title Key from the header and return it. Sites like this are called a decryption oracle. This principle is shown in figure 8. The way the AACS LA tracked down the attackers back then, was by making a phony disc header, which could be decrypted by half of the possible devices. They uploaded this header to the oracle and waited to see if it could extract the Title Key. This result let the AACS LA narrow down which device the oracle key might come from, and they continued this process until they found the oracles Device Key, which they then revoked. This principle is not very secure since there is a lot of way the attacker and the oracle can cheat the AACS LA. One of the ways to cheat the AACS LA, is by making the oracle pretend that is cannot extract the Title Key from the header information, which will slow down the AACS LA a little, but sooner or later the AACS LA will have found the Device Key. Another way to cheat the AACS LA is by limiting who can submit header information to the oracle. This will prevent the AACS LA by uploading their phony header.

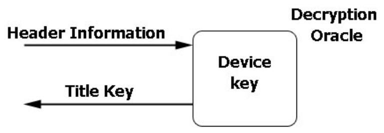


Figure 8: Illustration of decryption oracle

Because of the problems with the former version of Traitor Tracing the AACS LA added the Sequence Keys to the specification. The Traitor Tracing scheme used in the AACS is based on the pirated content itself or the keys to construct the content. The flowering scheme is based on the notes Jin et al. (2004), Jin et al. (2008) and Li (2008).This requires at good watermark scheme, which goal is to generate multiple versions of the same content, so that by getting a pirate copy, the AACS LA can use the embedded watermark to identify its origin version. In content traitor systems, assumptions about watermark robustness is very important, which is given any set of watermarked variations $t = v_1, v_2, \ldots, v_k$ for a segment, it is impossible to delete the watermark or generate another pirate variant $v_j | v_J \notin t$. This means that when a copy is found, it is possible to determine which variation of each content segment used in the copy, and thereby trace the device.

The key assignment process is divided up into two parts using respectively

[9]Halderman (Halderman)

an inner code and outer code. In the scheme used in AACS both the inner code and the outer code is Reed-Solomon. We will not go into further details about the Reed-Solomon error correcting code. The job of the inner code is to effectively create different movie versions. This means that it assign the different variations to a chosen point of the movie. F.eks. if there are 16 variations for each of the 15 movies, it could theoretically compute 16^{15} different versions, but since the scheme uses the Reed-Solomon code, it only creates 256 different versions. Even though this is much less than 16^{15}, any two versions will still differ at at least 14 points.

The outer code is used to assign movie versions to different players. As an example, each player is assigned one of the 256 versions for each movie (generated by the inner code) in a sequence of 255 movies. The Reed-Solomon code is capable of creating 256^4 code words (maximum numbers of players it can handle, and the largest number that can be represented by a 32-bit unsigned integer), and still have two different players differ at at least 252 movies. Together the inner code and the outer code form a super code, which is core in avoiding the space problem by having a small number of variations at any single point.

To sum up the idea behind the inner code and the outer code, the variation encrypting keys that are assigned using the inner code are embedded on the disc. The outer code uses the movie version keys to unlock a table (SKB) on the disc containing the actual variation encrypting keys for that movie. The movie versions keys are assigned the devices during manufacturing time, and are also what we know as sequence keys.

Tracing the compromised players is simple because of the way players are assigned the keys. I general the idea is to calculate the number of variations that a given player matches with the observed pirate movie. The scheme will then return its focus on the player who has the largest match, and revokes that player. Not only one plays can be excluded from the super code. This means the agency from a super code can exclude more players. This happens when more players have been used to create the pirate movie, from where the super code is derived.

To understand the principle a little better, we will use an example. Let us assume that we have a population of one billion players and we are using our 255 super codes. In this case any two players will differ from each other on at least 252 * 14 = 3528 variations over the entire sequence of movies. Remember that for each movie we have 256 encodings. This means that for any given movie, $\frac{1}{256}$ of the players encode the movie the same way. When given 3 movies, $(\frac{1}{256})^3$ of the players encodes those movies the same way. This means that given 4 movies, 0 players encodes these movies the same way, and the device is found. It is the same principle that is used when multiple devices has been used to create the pirate movie, but here we have to take into consideration that if N players are involved, $N + M$ innocent players could have produced the same result. Therefore more movies are needed to determine the involved players.

Having 9 movies the license agency knows that none of the pairs of players left is innocent, since to innocent players can at most produce 6 of the movies. Again a triplet of innocent players can produce the 9 movie, and each of the 3 players has 3 movies incoming with the guilty player. So the license agency can either chose to recover more movies or calculating the probability that an innocent triplet picked at random could have produced those nine movies.

We have now described how this efficient Traitor Tracing scheme works. But remember that nothing is stronger that the weakest link, so if the attacker was able to calculate the actual assignment for any other players, the pirate movie will looks as those it was produced by an innocent player. It is therefore important that this cannot happen.

5 Attacks on AACS

There are basically two different categories of attacks against the AACS. The first one contains the attacks which aim to decrypt the content so that it might be redistributed on the Internet, while the others are focused on copying the discs bit by bit and being able to play back the copies.

We will focus mostly on the first kind of attacks as they are the most relevant with respect to this course.

As described before there are two kinds of players that AACS supports, hardware and software players. The latter is the most vulnerable to attacks, as they do not take much effort to break into.

The method of getting information from a running process is called memory snooping, and this is the way hackers get access to the classified information stored in the memory space of the player. This method usually requires a debugger that is attached to the running process and thereby allowed to view or even modify the memory allocated to the process.

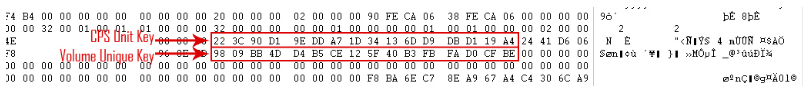
The only way to prevent the debugger to be attached is by making the computer supports Trusted Computing (TC) where it is predefined who has access to the process. On current hardware and software, Trusted Computing is not enforced and there are doubt that it will be in the future, due to controversy about the privacy and fearing giving the designers of TC systems too much power over the users.

The way programs are executed requires the data stored in the memory, to be able to perform operations to it. This includes the keys needed to obtain the decrypted content, and also the private host key used for the handshaking process between the player and the HD-drive to get the Volume ID of the media.

The idea is that hackers can get access to all the information that the player knows and thereby calculate the Title Keys used for decryption of the content, or even read them from the memory after the player has derived them.

The process of decrypting the content is simply to implement the AACS decryption protocol that is described in the specification.

The first known attack of this kind was done by muslix64, a consumer that brought an Xbox which supports HD-DVD playback. He discovered that due to lack of hardware support he was not allowed to get full access to the content. In his frustration he wrote a Java program by following the decryption protocol of AACS which could decrypt HD-DVDs and output .evo files that could be played on a PC. The software could not find the keys itself, but if the user submitted the right Title Keys, the program could successfully make a decrypted copy of the HD-DVD. The source code for program was released and soon a similar program for Blue-Ray was released, as well as programs that could extract the Title Keys from a running process of Cyberlink PowerDVD.



```
F4 B4 00 00 00 00 00  00 00 00 00 20 00 00 00  02 00 00 00 90 FE CA 06  38 FE CA 06 00 00 00 00 | 9ô´           þÊ 8þÊ
00 00 32 00 01 00 01  01 00 00 00 32 00 00 00  00 00 01 00 00 00 00 01  00 01 00 00 00 02 00 00 |   2        2
4E                    00 00 00 22 3C 90 D1  9E DD A7 1D 34 13 6D D9  DB D1 19 A4 24 41 D6 06 |  N  È       "<Ñ▮ÝS 4 mÙÛÑ ¤$AÖ
F8                    96 0E 0D 98 09 BB 4D  D4 B5 CE 12 5F 40 B3 FB  FA D0 CF BE 00 00 00 00 | Søn▮¢ù ´¥▮ }▮ »MÔµÎ _@³ûúÐÏ¾
00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |
00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  00 00 00 00 F8 BA 6E C7  8E A9 67 A4 C4 30 6C A9 |              ø°nÇ▮©g¤Ä0l©
```

Figure 9: Picture showing the CPS Unit Key = Title Key, and VUK located in the process memory of WinDVD. Source Jokin (2007, Doom9.org forums)

As a result of these programs, illegal copies of HD movies were soon to be found on various BitTorrent trackers even though muslix64 stated that the program should not be used for piracy.

Title Keys, Processing Keys and Volume IDs soon began showing up in various forums on the Internet. The response from the AACS LA was to send DMCA takedown notices to the sites posting the keys, as well as revoking the keys in the MKB. The administrators of the sites tried to delete the posts with the keys, but eventually gave up and decided to allow publication of the keys.

The first commercial program that allowed playback or copying of protected media was Slysoft AnyDVD. The program could find both the keys and decrypt the media. As it is a commercial program, the exact details on how the programs works is unknown, but users of the forum Doom9[10] claimed that the program uses a host certificate from PowerDVD.

All in all, it is just a race between the AACS LA to revoke the compromised keys and the hackers to find the keys in the newly released software players.

[10]evdberg (2007)

The other way of attacking AACS is by modifying the players so that they ignore the protection defined in the specification. A specific example of this is on the Xbox where users claim to have modified the firmware of the device to enable playback of copied HD-DVDs that do not have the right Volume ID.

5.1 BD+

As a new copy protection the BD+ was introduced. It developed by Macrovision and it was predicted that the scheme was to take 10 years to crack[11]. The BD+ offers extra security by having a virtual machine allowing content providers to include executable programs on the Blue-Ray discs. The programs can have different purposes like checking that the host environment is secure and that the player has not been tampered with.

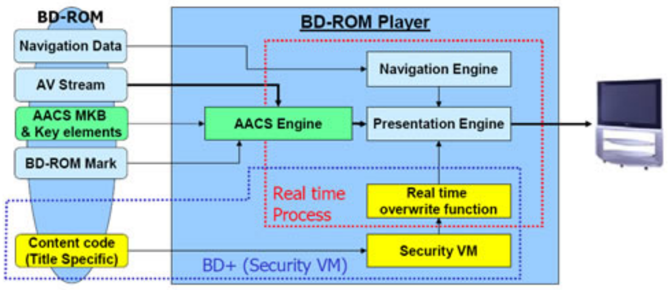


Figure 10: Illustration of the BD+ layer. Source Reimer (2007, Blu-ray content protection agency certifies BD+)

Even though the scheme was said to be secure for at least 10 years, the people at Slysoft began working on a version of AnyDVD which supported copying of BD+ media. And in March 2008 they circumvented the first version of BD+. Shortly after a new version of BD+ was released it was once again cracked by Slysoft.

6 Conclusion

After having read the specification of AACS and the articles about the different parts of the system, we got the impression that the system is secure.

The mix of encryption and authentication by using the AES algorithm for encryption and hashing and elliptic curves for authentication seemed to ensure that the system is bulletproof. The papers on the different parts of the systems, such as the Subset-Difference trees and the Traitor Tracking, have proofs that

[11](Singel, 2008)

the systems are secure and valid. We have not covered the mathematical proof in the security, since this would require much more space, and we really wanted to present the main principles in the AACS.

This impression changed radically when we read about the attacks against the AACS and how the attacks were performed. The Achilles' heel of the AACS is that even though the license agreement, for the manufactures of the players, states that the keys given to them by the AACS LA should be handled as highly confidential, the software players have shown not to comply with this agreement.

The most common way of retrieving the keys for unauthorized decrypting of AACS content, is by snooping the keys from the memory of a certified player.

However, the blame should not be entirely on the software developers as there currently is no support for Trusted Computing Base in the most popular operating systems.

So we can end this conclusion by saying that not even AACS is stronger that the weakest link, and in this case the weakest link is the software.

References

evdberg (2007, February). Anydvd method of operation. http://forum.doom9.org/showthread.php?t=122272.

Halderman, J. A. Aacs: Blacklisting, oracles, and traitor tracing. http://freedom-to-tinker.com/blog/jhalderm/aacs-blacklisting-oracles-and-traitor-tracing.

Intel, I. B. Machines, Microsoft, Panasonic, Sony, Toshiba, T. W. Disney, and W. Bros (February 24, 2009b). *Pre-recorded Video Book*. AACS LA.

Intel, I. B. Machines, Microsoft, Panasonic, Sony, Toshiba, T. W. Disney, and W. Bros (May 21, 2009a). *Introduction and Common Cryptographic Elements Book*. AACS LA.

Jin, H., J. Lotspiech, and N. Megiddo (2008). Efficient coalition detection in traitor tracing. In *IFIP International Federation for Information Processing*, pp. 365–379.

Jin, H., J. Lotspiech, and S. Nusser (2004). Traitor tracing for prerecorded and recordable media. In *DRM '04: Proceedings of the 4th ACM workshop on Digital rights management*, New York, NY, USA, pp. 83–90. ACM.

Jokin (2007, January). Post blu-ray volume unique keys here. http://forum.doom9.org/showthread.php?t=120988.

Li, C.-t. (2008). *Multimedia Forensics and Security*.

Naor, D., M. Naor, and J. Lotspiech (2001). Revocation and tracing schemes for stateless receivers. In J. Kilian (Ed.), *CRYPTO*, Volume 2139 of *Lecture Notes in Computer Science*. Springer.

Reimer, J. (2007, June). Blu-ray content protection agency certifies bd+. `http://arstechnica.com/security/news/2007/06/blu-ray-content-protection-agency-certifies-bd.ars`.

Singel, R. (2008). How crypto won the dvd war. `http://www.wired.com/threatlevel/2008/02/how-crypto-won/`.

Wikipedia, t. f. e. (2009a, December). Aacs encryption key controversy. `http://en.wikipedia.org/wiki/AACS_encryption_key_controversy`.

Wikipedia, t. f. e. (2009b, December). Advanced access content system. `http://en.wikipedia.org/wiki/Advanced_Access_Content_System`.

Wikipedia, t. f. e. (2009c, November). Security of advanced access content system. `http://en.wikipedia.org/wiki/History_of_attacks_against_Advanced_Access_Content_System#History_of_attacks`.

Wikipedia, t. f. e. (2009d, December). Vhs. `http://en.wikipedia.org/wiki/VHS`.