An Overview of the Advanced Access Content System (AACS)

Kevin Henry, Jiayuan Sui, Ge Zhong

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1, Canada
{k2henry, jsui, gzhong}@cs.uwaterloo.ca

Abstract

The Advanced Access Content System (AACS) is a content distribution system for recordable and pre-recorded media, currently used to protect HD-DVD and Blu-Ray content. AACS builds off of its predecessors, the Content Scramble System (CSS) and Content Protection for Pre-recorded/Recordable media (CPRM/CPPM), adding more robust key distribution and revocation capabilities. In this paper, a concise summary of the lengthy AACS specification for pre-recorded media is provided, with emphasis on how the AACS keys are organized, distributed, and revoked using subset difference trees. A description of the AACS mechanisms for recordable content and on-line content is also provided.

1 Introduction

The Advanced Access Content System (AACS) is a content distribution system for recordable and pre-recorded media. Most notably, the AACS is used to protect high definition video content on Blu-Ray and HD-DVD discs. The current AACS technical specifications span nearly 550 pages spread across seven documents [3-9], and provide an overview of the system, as well as specific implementations for pre-recorded and recordable media for both Blu-Ray and HD-DVD formats. The goal of this paper is to provide a concise summary of the information contained within these specifications, with extra attention given to the broadcast encryption scheme used. Over the entire specification, only two or three pages are dedicated to how a device's cryptographic keys are distributed and revoked, however this mechanism, described in Section 2.3, is the core upon which the rest of the system is built. In place of the brief summary in the technical specifications, we provide a thorough explanation of the theory behind the key distribution and key revocation used in the AACS.

We follow the convention of the AACS specification and capitalize the AACS specific terms such as Media Key Block, Title Key, etc.

1.1 Background

AACS is the successor to the Content Scramble System (CSS) [2] used to protect the content on DVDs, and the Content Protection for Recordable/Pre-Recorded Media (CPRM/CPPM) scheme [1]. The Content Scramble System is based on a proprietary stream-cipher that was cracked in 1999, shortly after the DVD format began gaining popularity. Because CSS provides no mechanism for revoking a compromised key the system has been completely broken since the initial compromise.

CPRM/CPPM was built to be more robust than its predecessor, incorporating key revocation for compromised devices and traitor tracing capabilities, and is mostly used in flash memory devices such as Secure Digital (SD) memory cards. AACS incorporates the same traitor tracing scheme as the CPRM/CPPM, allowing a compromised device to be located even if the compromised key is not made public.

1.2 Organization

Section 2 introduces the broadcast encryption scheme used in the AACS, based on subset difference trees, followed by a description of how content is encrypted and decrypted under this scheme for pre-recorded media in Section 3. A slightly different process is used for recordable media, which is explained in Section 4. The paper concludes with a brief discussion of the authentication protocols used by the AACS in Section 5 and its on-line capabilities in Section 6, followed by a discussion of future work.

2 Broadcast Encryption

2.1 Overview

Broadcast encryption is gaining increasing popularity in commercial applications. This cryptographic tool is used by a broadcast center (or trust authority) to distribute an encrypted message to a set of users such that only qualified users (e.g. subscribers who have paid the fee and behaved according to the rules) can decrypt the encrypted message, but no others can.

Let us use $\mathcal{U}$ to denote the set of n users in a broadcast encryption scheme,

$$\mathcal{U} = \{u_1, u_2, \ldots, u_n\}.$$

The encrypted message is broadcast to $\mathcal{U}$ by the broadcast center. Only a subset of $\mathcal{U}$, denoted $\mathcal{P}$, can decrypt the encrypted message. We refer to $\mathcal{P}$ as the set of privileged users. The users in $\mathcal{U}$ that can not decrypt the encrypted message comprise the set of revoked users, denoted by $\mathcal{R}$. Clearly, $\mathcal{P} = \mathcal{U} \setminus \mathcal{R}$, $\mathcal{P} \cup \mathcal{R} = \mathcal{U}$, and $\mathcal{P} \cap \mathcal{R} = \emptyset$.

Most broadcast encryption schemes, including the scheme used in the AACS, can be divided into three steps:

(1) An initialization step assigns each user in $\mathcal{U}$ some secret information (e.g. a set of keys) which allows the user to decrypt the encrypted message.

(2) The broadcast center uses a broadcast encryption algorithm which takes as input a message $\mathcal{M}$ and a set of users $\mathcal{R}$ whose decryption privileges are revoked, and outputs an encrypted message. The encrypted message is then broadcast to $\mathcal{U}$.

(3) Upon receiving the broadcast, a user can use her secret information to decrypt the encrypted message to obtain $\mathcal{M}$ if and only if she is in $\mathcal{P}$.

Since this broadcast encryption model is a function that takes as an input a set of revoked users $\mathcal{R}$, it is also referred to as *Revocation Scheme*.

The revocation scheme used by the AACS assumes "stateless receivers". In such a scheme, receivers are "off-line", which means that once the secret information is assigned to each receiver, it becomes very hard to update. On the other hand, "stateful receivers" are much more flexible, as they

can update their secret information regularly with ease, however, this might require a continuous on-line connection. The merit of the "stateless receivers" model is that it allows the hardware/software at the receiver end to be as simple as possible.

2.2 Subset-Cover Framework

Naor et al. have suggested a subset-cover framework [17] which serves as a nice prototype for most revocation schemes, including the one employed by the AACS. In this framework, a set of non-empty subsets of $\mathcal{U}$, $S = \{S_1, S_2, \ldots, S_w\}$, $S_j \subseteq \mathcal{U}$, is defined such that $\mathcal{U} = \bigcup_{j=1}^{w} S_j$. Subsets need not be disjoint, and a single user may be in many subsets. A long-lived key L_j is assigned to each subset S_j. A user u in S_j should be able to deduce L_j easily from her secret information. If S is constructed in an appropriate way, then for any non-empty set of privileged users $\mathcal{P}$ we can find a set of subsets $C = \{S_{i_1}, S_{i_2}, \ldots, S_{i_m}\}$ that "covers" $\mathcal{P}$, i.e.

$$\mathcal{P} = \mathcal{U} \setminus \mathcal{R} = \bigcup_{j=1}^{m} S_{i_j}.$$

Note that $C \subseteq S$, and is called a subset-cover for $\mathcal{P}$. We can then encrypt a session key, K, m times with $L_{i_1}, L_{i_2}, \ldots, L_{i_m}$, where K is used to encrypt the message M.

As we have already mentioned in the previous section, there are three steps in a revocation scheme, namely, initialization, broadcast encryption, and broadcast decryption. The subset-cover framework follows these three-steps:

Initialization: A user $u \in \mathcal{U}$ is given a piece of secret information I_u by the broadcast center. I_u is used to deduce the long-lived key L_j corresponding to the subset S_j in which she is included. If a user is in many subsets, then I_u should be able to allow her to deduce all the long-lived keys for those subsets. If each of the keys L_j are generated uniformly at random and independently from each other, we say that the keys are chosen information-theoretically. On the other hand, if the keys are obtained as a function of some secret information, then the keys are generated computationally. Using computational key generation instead of information-theoretic key generation can greatly reduce the space requirements for key storage, however, it sacrifices information-theoretic security for computational security. In this framework, I_u can be a set of long-lived keys $\{L_{u_1}, L_{u_2}, \ldots, L_{u_k}\}$ (information-theoretic), or it can be an input to a function which outputs a set of keys (computational).

Broadcast encryption: Given a message M and a set of revoked users $\mathcal{R}$, the broadcast center first chooses a session key K with which to encrypt M. The center then finds a subset-cover $C = \{S_{i_1}, S_{i_2}, \ldots, S_{i_m}\}$ which covers the set of non-revoked users $\mathcal{P}$. The long-lived keys $\{L_{i_1}, L_{i_2}, \ldots, L_{i_m}\}$ corresponding to the subsets in the subset-cover are then used to encrypt the session key. The final broadcast made by the broadcast center to $\mathcal{U}$ is

$$\langle \underbrace{i_1, i_2, \ldots, i_m, E_{L_{i_1}}(K), E_{L_{i_2}}(K), \ldots, E_{L_{i_m}}(K)}_{header}; E_K(M) \rangle.$$

The portion that precedes the semicolon is called the header of the broadcast, and $E_K(M)$ is termed the body of the broadcast. The size of the header is referred to as the broadcast message expansion.

Broadcast decryption: Upon receiving the broadcast, a user u can first search through the list of indices, $\{i_1, i_2, \ldots, i_m\}$, to find an index i_j such that $u \in S_{i_j}$. If such an index can not be

found, then the user is revoked. The user u then extracts the corresponding key L_{i_j} from I_u and uses L_{i_j} to decrypt the corresponding encrypted session key, $D_{L_{i_j}}(E_{L_{i_j}}(K)) = K$. Finally, u uses K to obtain the message, $D_K(E_K(M)) = M$.

To illustrate how revocation schemes can be built based on this framework, we present two trivial revocation schemes, A and B, along with some comments on their impracticalities.

Trivial A: Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be a set of users. The broadcast center constructs S in a way such that each subset contains only one user, i.e. $S = \{\{u_1\}, \{u_2\}, \ldots, \{u_n\}\}$. A long-lived key is assigned to each subset, which is to say that each user u_i is given one long-lived key L_i. Whenever the broadcast center wants to send a message to a set of privileged users $\mathcal{P}$, it encrypts the message using a session key and then encrypts the session key with the key L_i, for all i such that $u_i \in \mathcal{P}$. Upon receipt of the broadcast, a privileged user uses her long-lived key to compute the session key and then uses the session key to obtain the message.

A nice property of this scheme is that each user is required to store only a single key, hence the storage size is optimal. However, the message size grows linearly with respect to the size of $\mathcal{P}$. If the broadcast center needs to send a message to one million recipients, the message expansion would be two million, which in most cases is an unrealistic requirement.

Trivial B: Again, let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be the set of users. The set S is constructed such that it includes every possible non-empty subset of $\mathcal{U}$, i.e. $S = 2^{\mathcal{U}} \setminus \emptyset$. A long-lived key is assigned to each subset, hence there are $2^n - 1$ long-lived keys in total. When the broadcast center sends a message to a set of privileged users $\mathcal{P}$, it finds the subset S_j that is equal to $\mathcal{P}$ and encrypts the session key using the long-lived key L_j corresponding to S_j. The final broadcast is simply $\langle j, E_{L_j}(K); E_K(M)\rangle$. The decryption process for the users in $\mathcal{P}$ should be straightforward.

This scheme is essentially the opposite of the previous scheme. The message expansion is always 2 which is optimal, however each user must store a large number of keys. In the case of $|\mathcal{U}| = n$, each user must store

$$1 + \binom{n-1}{1} + \binom{n-1}{2} + \ldots + \binom{n-1}{n-1} = \sum_{k=0}^{n-1} \binom{n-1}{k} = 2^{n-1}$$

keys. Thus, with only a few dozen users this scheme becomes completely impractical.

There is always a trade-off between broadcast message expansion and the number of keys the user is required to store. The challenge is to find a revocation scheme that gives a proper balance between the two. A fairly efficient revocation scheme (used by the AACS) is discussed in the next section.

2.3 Subset Difference Revocation Scheme

Naor et al. have proposed the following subset difference revocation scheme (SDRS) along with the subset-cover framework which was introduced in the previous section [17]. A Ph.D thesis by Martin also explains SDRS in detail [16].

2.3.1 Basic Scheme

The SDRS is a tree-based revocation scheme. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ be a set of n users where $n = 2^h$, such that each user corresponds to a leaf node in a complete binary tree of height h. If $n \neq 2^h$, then we can use the smallest complete binary tree with more than n leaves. The complete

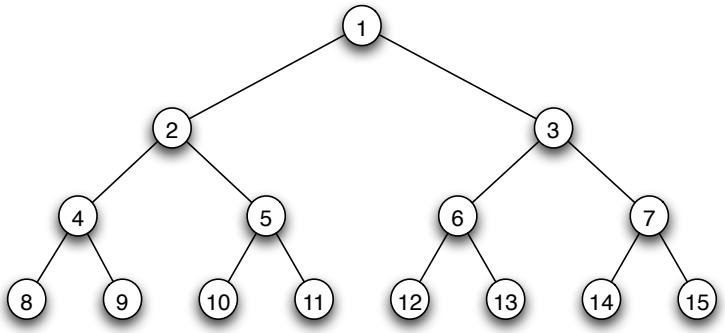

Figure 1: A complete binary tree with 8 leafs.

binary tree has $2n-1$ nodes, labeled breadth-first for v_i, $i = 1 \ldots 2n-1$. Figure 1 shows a complete binary tree with eight leaves where nodes v_8 through v_{15} correspond to eight users.

We form S to be a set of differences between two subsets of $\mathcal{U}$. A difference between two subsets of $\mathcal{U}$ is denoted $\gamma(v_i, v_j) = desc(v_i) \setminus desc(v_j)$, where v_j is a descendant of v_i and $desc(v_i)$ is the set of leaf nodes in the subtree rooted at v_i. There is a special case: $\gamma(v_0, v_0) = \mathcal{U}$. Clearly, every $\gamma(v_i, v_j)$ is a subset of $\mathcal{U}$. Figure 2 gives an example of $\gamma(2, 5)$. Hence,

$$S = \{\gamma(v_i, v_j) \mid 1 \leq i, j \leq 2n-1 \text{ and } desc(v_j) \subseteq desc(v_i) \text{ or } (i, j) = (0, 0)\}.$$

In the case of information-theoretic key assignment, each element in S, $\gamma(v_i, v_j)$, is assigned a long-lived key, $L_{i,j}$, and each user in S has a copy of $L_{i.j}$. Let $\mathcal{R}$ be a set of revoked users, a fairly straightforward algorithm is given to find the subset-cover for $\mathcal{U} \setminus \mathcal{R}$:

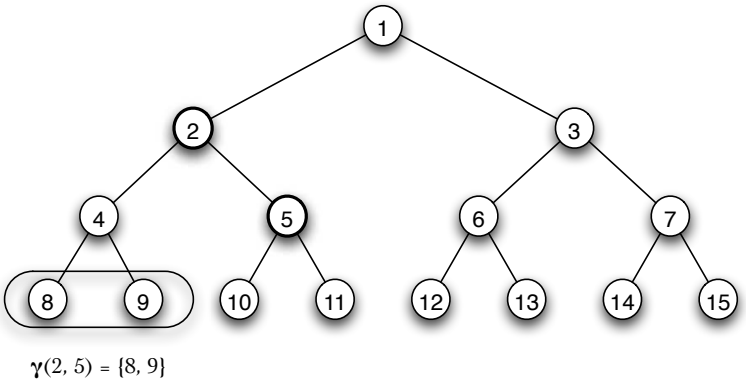

Figure 2: A set of leaf nodes represented by $\gamma(2, 5)$.

Algorithm 1: Subset-Cover Algorithm

Initialize the subset-cover C to an empty set, and let $ST(R)$ be the Steiner tree induced by $\mathcal{R}$ and the root v_1

while *$ST(R)$ has more than one leaf* **do**

Find two leaves v_i and v_j such that their least common ancestor, v, has no other leaves in $ST(R)$

Let v_k and v_l be children of v such that v_i is a descendant of v_k and v_j is a descendant of v_l

if $v_k \neq v_i$ **then**
$C = C \cup \{\gamma(v_k, v_i)\}$
else if $v_l \neq v_j$ **then**
$C = C \cup \{\gamma(v_l, v_j)\}$
end

Remove everything descended from v in $ST(R)$ and make v a leaf node of the new $ST(R)$

end

if *the only leaf v_i of $ST(R)$ is not the root v_1* **then**
$C = C \cup \{\gamma(v_1, v_i)\}$
end

After executing this algorithm, $C = \{\gamma(v_{i_1}, v_{j_1}), \ldots, \gamma(v_{i_m}, v_{j_m})\}$ is the subset-cover for $\mathcal{U} \setminus \mathcal{R}$. Upon finding C, the broadcast center can simply create the broadcast

$$\langle (i_1, j_1), \ldots, (i_m, j_m), E_{L_{i_1,j_1}}(K), \ldots, E_{L_{i_m,j_m}}(K); E_K(M) \rangle$$

and send it to $\mathcal{U}$. Figure 3 shows an example in which three leaf nodes are revoked, and $ST(R)$ is highlighted. Using the above algorithm, we find that the subset-cover for the remaining leaf nodes is $\{\gamma(2, 8), \gamma(3, 6)\}$.

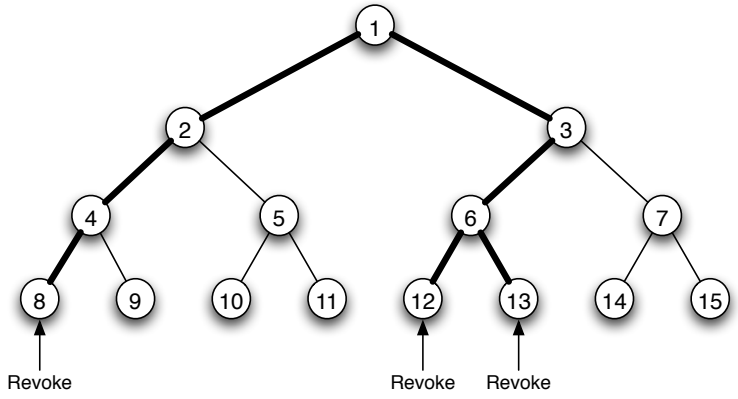

Figure 3: A Steiner subtree induced by the root and the three revoked leafs.

2.3.2 Analysis

(1) **Storage Complexity at the Server End:**

The size of S, which is the number of all possible subsets of $\mathcal{U}$ formed by the subset difference method, can be calculated in the following way: The root of a tree can form a subset difference with every one of its descendants. A subtree of a complete binary tree T rooted at v_i at depth k has $\lfloor (2n-1)/2^k \rfloor$ nodes, where n is the number of leaves in T, so v_i can form $\lfloor (2n-1)/2^k \rfloor - 1$ subset differences with all its descendants. Calculating all the possible subset differences in T is equal to calculating the sum of every non-leaf node's descendants:

$$\sum_{k=0}^{(\log_2 n)-1} 2^k (\lfloor \frac{2n-1}{2^k} \rfloor - 1),$$

which is the same as summing all the nodes in T below depth k a total of $\log_2 n$ times, with each iteration incrementing the depth by 1:

$$\sum_{k=0}^{(\log_2 n)-1} (2n - 1 - (2^{k+1} - 1)) = \sum_{k=0}^{(\log_2 n)-1} (2n - 2^{k+1}).$$

The maximum value for k is $(\log_2 n) - 1$, so the biggest value for 2^{k+1} is $2n - n = n$. Hence, the smallest term in the summation is n. The summation adds up $\log_2 n$ terms, $n \leq \text{term} < 2n$, so the overall value is bounded by $2n \log_2 n \in O(n \log n)$.

If the long-lived keys are assigned information-theoretically for every subset in S, then $O(n \log n)$ is the data storage complexity at the server side (i.e. broadcast center). Although a vast improvement over the worst case, this is still not sufficient for a system the size of the AACS. For example, if T is of height 31 and each long-lived key is of size 128 bits, the server must store more than a terabyte of long-lived keys securely.

(2) **Data Complexity of the Message Expansion:**

In Algorithm 1, each iteration through the while loop increases the number of subsets in C by at most 2, while reducing the number of Steiner leaves by 1. At the final step (step 3), when there is only one leaf, at most one subset is added to C. Hence, given any set of revoked users, $\mathcal{R}$ with cardinality r, the size of the subset-cover for $\mathcal{U} \setminus \mathcal{R}$ is at most $2r - 1$. Thus, the data complexity of the message expansion is $O(r)$.

(3) **Storage Complexity at the Client End:**

We now determine how many long-lived keys a user, u, is required to store, i.e., how many subset-differences is u a member of. Suppose the tree T containing u has n leaves. We start by considering the subtree of T of height h that contains u. There are $2^{h+1} - 1$ nodes in this subtree and $h+1$ nodes on the path between u and the subtree's root, v_i. Only these nodes can not be used to form subset differences with v_i, because otherwise u would be excluded. Hence, $2^{h+1} - 1 - (h+1)$ subset differences containing u can be formed with v_i in this subtree. u is in

$\log_2 n$ subtrees in T, so the total number of subset differences containing u is

$$\begin{aligned} & \sum_{h=1}^{\log_2 n} (2^{h+1} - 1 - (h+1)) \\ = & \sum_{h=1}^{\log_2 n} 2^{h+1} - \sum_{h=1}^{\log_2 n} h - \sum_{h=1}^{\log_2 n} 2 \\ = & (2^{\log_2 n+2} - 4) - \frac{\log_2 n(\log_2 n + 1)}{2} - 2\log_2 n \\ = & 4n - \frac{\log_2^2 n}{2} - \frac{5\log_2 n}{2} - 4. \end{aligned}$$

When n is large, the above number is roughly equal to $4n$. If we use information-theoretic key assignment, each user has to store as many long-lived keys as the number of subset differences she is in, which means that a user has to store around $4n$ keys. This is undesirable for larger systems. For example, if a system has 2^{31} users and each long-lived key has size 128 bits, then each user has to store over a hundred gigabytes of long-lived keys securely.

In the next section, we introduce a computational key assignment method to greatly reduce the key storage requirement at both the server end and the client end.

2.3.3 Computational Key Assignment

Up until this point, all long-lived keys in the subset difference revocation scheme have been chosen uniformly at random and independently from each other. As we have seen, this information-theoretic key assignment induces great storage overhead for both the broadcast center and the user. To reduce storage complexity we can instead use computational key assignment, which means that the long-lived keys are calculated from some secret information.

The idea is that instead of giving the user long-lived keys, a number of "secret labels" are given to the user, such that many long-lived keys can be calculated from a single label. For each pair of nodes (v_i, v_j) such that v_j is a descendant of v_i, there is a label, $LABEL_{i,j}$ associated with it. The long-lived key $L_{i,j}$ for $\gamma(v_i, v_j)$ as well as all the labels for node pairs (v_i, v'_j), where v'_j is a descendant of v_j, can be calculated from $LABEL_{i,j}$. To implement this idea, a pseudo-random sequence generator is used.

Let function, $G : \{0,1\}^a \rightarrow \{0,1\}^b$, where $b = 3a$, be a pseudo-random sequence generator such that the size of the output is 3 times the size of its input. A necessary condition for the pseudo-random sequence generator G is that given a randomly chosen input string no polynomial-time adversary can distinguish the output of G from a random sting of the same length. G takes a label as its input, and outputs

$$G(LABEL_{i,j}) = LABEL_{i,2j}||L_{i,j}||LABEL_{i,2j+1},$$

where "||" is the concatenation operator, $L_{i,j}$ is the long-lived key for $\gamma(v_i, v_j)$, and the two newly generated labels are for (v_i, v_{2j}) and (v_i, v_{2j+1}). Since we are using breadth-first labeling for the nodes, v_{2j} and v_{2j+1} are the two children of v_j. Figure 4 gives a pictorial example to show that many secret labels, and hence many long-lived keys, can be generated from a single label.

A user u is given the labels $LABEL_{i,j}$, for all i such that v_i is on the path from u to the root (u excluded). The v_js are the nodes that "hang off" the path from v_i to u (u excluded). With this setup, all long-lived keys required by u can be generated from the set of labels. For v_i at height h,

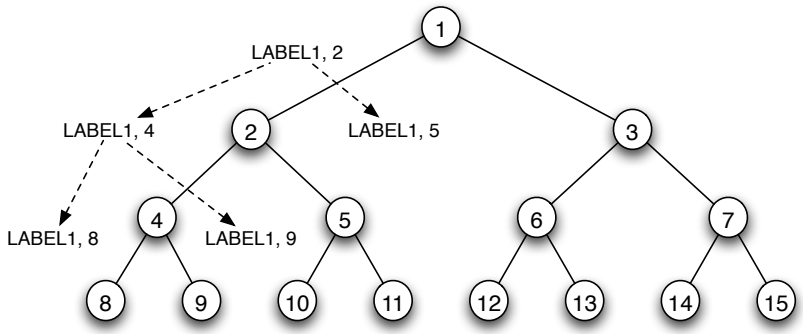

Figure 4: Many secret labels can be generated from a label.

there are h different v_j's "hanging off" the path from v_i to u, hence the total number of labels held by u is

$$\sum_{i=1}^{\log_2 n} i = \frac{((\log_2 n) + 1)\log_2 n}{2} = \frac{1}{2}\log_2^2 n + \frac{1}{2}\log_2 n.$$

We also need to include the key for the case when there is no revocation, which we take to be the long-lived key $L_{0,0}$. Thus, the total number of secret values a user must store is

$$\frac{1}{2}\log_2^2 n + \frac{1}{2}\log_2 n + 1.$$

Hence, we obtain storage complexity $O(\log^2 n)$ at the client end using computational key assignment, as opposed to $O(n)$ using information-theoretic key assignment. For the same example from the last section where a system has 2^{31} users and each label/long-lived key has size 128 bits, u is only required to store around 8 kilobytes (instead of more than one hundred gigabytes) of key information securely using the computational key assignment approach.

The storage requirement for the broadcast center is greatly reduced as well. The broadcast center only requires $LABEL_{i,i}$ for each non-leaf node v_i in order to generate all the labels required by the users. So, for a tree with n leafs, the broadcast center must store $n-1$ labels. Computational key assignment reduces the storage requirement for the server to $O(n)$ from $O(n \log n)$ required by information-theoretic key assignment. Continuing with the same example, the server is required to store about 32 gigabytes (instead of over 1 terabyte) of secret key information for computational key assignment. Although this is an improvement, storing 32 gigabytes of data securely still places a great deal of stress on the server end. Further improvements are possible by generating the labels in a non-information-theoretic manner. For example, instead of storing each label, the server can generate the label for any node using a keyed pseudo-random function which takes as an input the name of the node. This reduces the storage complexity at the server end to O(1), as only the key for the pseudo-random function must be stored.

In the next section we discuss how the subset difference key revocation scheme is used in the AACS to recover the necessary information to decrypt encrypted content in the pre-recorded setting.

3 Encryption and Decryption of Pre-Recorded Media

Pre-recorded media refers to any media that is distributed with content pre-loaded, although in current implementations it generally refers to HD-DVD and Blu-Ray discs pre-recorded with video, games, or software. On the other hand, recordable media usually refers to the set of blank HD-DVD and Blu-Ray discs on which consumers can record their own data. For the ease of explanation, we assume that optical discs are the storage medium being used.

Content encryption is implemented by the replicator, while decryption is performed by the playback device at the client end. Figure 5 shows an overview of the content encryption and decryption processes. Since we have not yet gathered enough information about Sequence Keys, a special set of keys used for traitor tracing, we do not take them into consideration. An overview of the basic technique used for traitor tracing is given in Section 7, although specific implementation details are still under investigation.

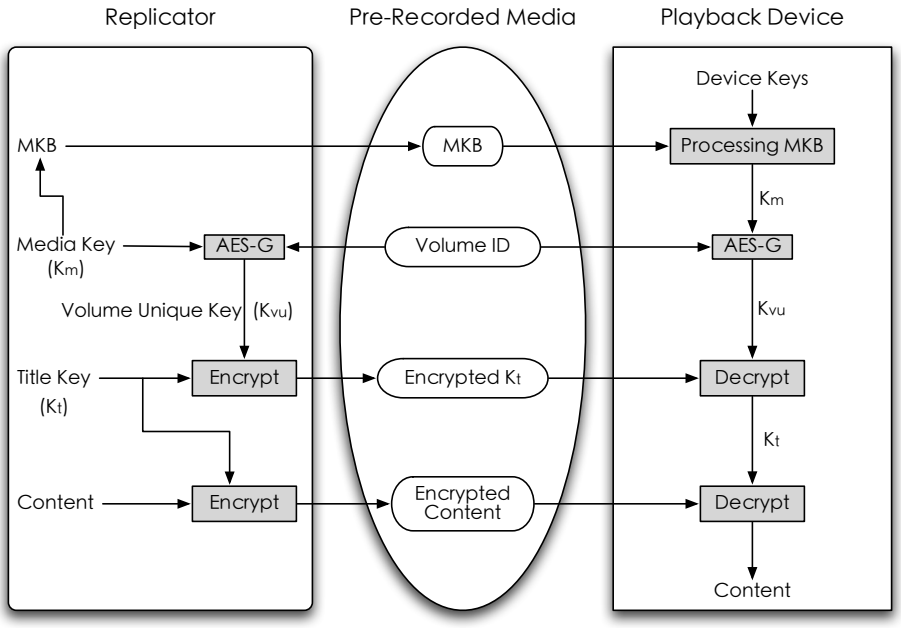

Figure 5: Encryption and Decryption Overview for Pre-recorded Media.

Before explaining the procedures for content encryption and decryption, we first introduce some terms and cryptographic elements used in the AACS.

Title: The content owner, e.g. a Hollywood movie studio, provides the content to a licensed replicator in the form of one or more Titles. For example, a movie, such as King Kong, is called a Title.

Title Key ($\mathbf{K_t}$): The replicator randomly generates a secret 128-bit Title Key for each Title. At the replicator's discretion, the Title Key can be used to encrypt all instances of a given Title, or different Title Keys may be used for different instances. So far all instances of a single Title have been encrypted with the same Title Key. The uses of Title Keys are explained in more detail in Section 6

Volume ID: The replicator also randomly generates an 128-bit identifier for each title, called the Volume Identifier (Volume ID), which is then burned onto the disc in a special area called the "burst cut area". This special area on either pre-recorded or recordable discs is read-only for consumer recorders. The purpose of the Volume ID is to protect against "bit-by-bit copying" of protected content. As with the Title Key, the same Volume ID can be used for all instances of a given Title, or different Volume IDs may be used for different instances.

Media Key ($\mathbf{K_m}$): The Media Key is provided by the AACS Licensing Authority (the AACS LA), and is used together with the Volume ID to compute the Volume Unique Key. As with most the AACS keys, the Media Key has length 128 bits.

Volume Unique Key ($\mathbf{K_{vu}}$): Volume Unique Key is used to encrypt the Title Key. It is the output of a one-way function, AES-G, which takes as inputs the Volume ID and the Media Key.

AES: Advanced Encryption Standard (AES) is the building block for data encryption/decryption in the AACS. The content is encrypted/decrypted using AES in CBC mode. The various keys are encrypted/decrypted using AES in ECB mode. Since most of the keys are of size 128 bits, one AES encryption/decryption is sufficient for the encryption/decryption of keys.

AES-G: AES-G is an AES-based one-way function that takes two inputs of length 128 bits, and produces an output of size 128 bits. The AES-G function is depicted in Figure 6.

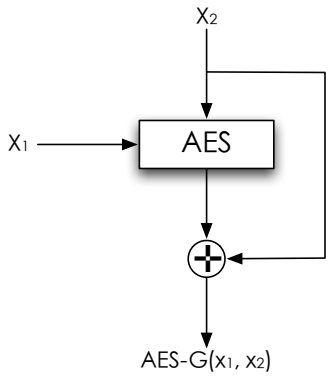

Figure 6: AES-based one-way function AES-G.

Device Key: Device Keys serve the same purpose as labels in the subset difference revocation scheme. Each has length 128 bits and is stored in the playback device.

Processing Key: Processing Keys are the equivalent of long-lived keys in the subset difference revocation scheme. Each has length 128 bits and can be derived from the Device Keys. Processing Keys are used to encrypt the Media Key.

Media Key Block (MKB): The Media Key Block is provided by the AACS LA, and contains lists of information about revoked Hosts and Drives. The Media Key is encrypted with one or more Processing Keys and placed in the Media Key Block along with the indices of the Processing Keys used in the encryption (these together are equivalent to the header of the broadcast in the subset difference revocation scheme).

Content Encryption:

1. The content owner gives the content to the licensed replicator, who randomly generates a secret Title Key for the content. The content is then encrypted with the Title Key using AES in CBC mode.

2. For each Title (or a set of Titles), the AACS LA gives a MKB and a secret Media Key to the replicator. The Encrypted Media Keys are included in the MKB, and only privileged users are able to decrypt one of them to recover the Media Key.

3. The replicator randomly generates a Volume ID for the content. It then uses the Volume ID and together with the Media Key to calculate the Volume Unique Key using the AES-G function. The Title Key is encrypted with the Volume Unique Key using AES.

4. Finally, the replicator places the MKB, Volume ID, Encrypted Title Key, and encrypted content onto the disc, along with several other pieces of information not directly related to content encryption. Note that the Volume ID is placed inside the "burst cutting area" on the disc and is read-only for all consumer recorders.

Content Decryption:

1. If the playback device is not revoked, then using the subset difference revocation scheme it should be able to use its Device Keys to derive a Processing Key which can decrypt one of the Encrypted Media Keys recorded in the MKB.

2. The playback device then reads the Volume ID off the disc and inputs it together with the Media Key into the AES-G function to calculate the Volume Unique Key.

3. Upon obtaining the Volume Unique Key, the Title Key is recovered by decrypting the Encrypted Title Key with it.

4. Finally, the playback device decrypts the encrypted content using the Title Key and plays the content.

According to the technical specifications [7], each device stores 253 Device Keys. Recall that a Device Key is the same as a label in the subset difference revocation scheme. If a user stores 253 labels, it implies that the height of the complete binary tree is 22. However, we have not yet found any authoritative information regarding the size of the complete binary tree used in the AACS (it is vaguely mentioned in a paper written by one of the designers of the AACS [15]).

There is a fairly straightforward authentication between the disc and the playback device. The replicator produces a hash of the encrypted content and gives it to the AACS LA to sign it using the entity's private key. The signature is called content certificate. The content hash and the content certificate are put onto the disc. The playback device has the AACS LA's public key, so it can verify the content certificate. If the verification fails, the player should not allow content playback. The player also produces a hash of the encrypted content, and this hash is compared against the content hash on the disc. If the comparison returns false, the player should prevent playback of the content. We take a closer look at authentication in the AACS in Section 5.

3.1 Real-World Security of the AACS

It would seem that the encryption protocol used by the AACS is robust, but this begs the following question: Why have so many HD-DVD and Blu-Ray been pirated? The reality is that attackers have been able to recover various keys used in the encrypting process of the content. Since December 26th, 2006, attackers have managed to locate the Title Key for any Title by taking memory dumps from a Windows-based HD-DVD/Blu-Ray software decoder called WinDVD. Once a Title Key is compromised, anyone who has the corresponding Title can decrypt the encrypted content freely, bypassing the usual the AACS key recovery. On February 13th, 2007, attackers recovered a Processing Keys for both HD-DVD and Blu-Ray discs in another memory dump of WinDVD. Recall that the Processing Key is equivalent to the long-lived key in the subset difference revocation scheme. At the time of the compromise, no devices had been revoked, so all Media Keys for all HD-DVD and Blu-Ray discs were encrypted with these two compromised Processing Keys. Knowledge of these keys allows all discs released prior to revocation to be decrypted. On February 27th, 2007, attackers managed to extract some of the Device Keys stored in WinDVD.

On April 6th, 2007, the Device Keys for WinDVD were finally revoked. The AACS LA has blamed poor software implementation for the compromise, rather than a weakness in the system itself. Nevertheless, the leaking of just two Processing Keys has allowed all HD-DVD and Blu-Ray movies released before April 2007 (the first HD-DVD and Blu-Ray titles were released in April 2006 and June 2006 respectively) to be copied freely.

4 Recordable Media

In addition to protecting pre-recorded media, the AACS also provides mechanisms for protecting media added to a recordable disc after manufacturing. This is made possible by logically separating a disc into three pieces.

- **Read-only Area:** A portion of the disc that cannot be written by consumer devices. The read-only area is used to store the *Media Identifier*, a 128-bit value that is unique to each disc. By including a unique identifier on each disc that cannot be modified by consumer devices, it becomes impossible to create a truly identical copy of the disc. The Media Identifier is used to calculate a MAC with the Title Key, allowing the drive to detect whether or not the Title Key was generated for this specific disc, or inappropriately copied from another. The read only area may also contain a Media Key Block, which is used for distributing updated MKBs to the read/write area described later in this section.

- **Protected Area:** A special area that can only be accessed and written by the drive. The host, such as a piece of software, cannot instruct the drive to write a specific value to the protected area. Instead, the drive is capable of independently generating and writing the appropriate information to the protected area when necessary. The only piece of information stored in the protected area is the *Binding Nonce* (or nonces), a special value which is used to retrieve Title Keys on recordable media.

- **Read/Write Area:** The area where the encrypted content is written, along with its associated Media Key Block, Encrypted Title Keys, and Usage Rules. Each piece of encrypted information written to the disc contains its own MKB and Title Key File (containing the encrypted Title Keys associated with the content), as well as its own Usage Rules. The Usage Rules for a given piece of content are generated by the content owner and include information about who may access the content and what restrictions for copying are in place. The Usage Rules are also

included in the decryption process in a way that prevents an unauthorized party from altering them.

In general, there are two types of recordable media, re-writable and write-once. In the case of write-once media, updating a piece of written content is impossible, so the protocols described in this section may only be carried out once.

4.1 Media Key Block

As mentioned previously, each piece of encrypted content has a corresponding Media Key Block written in the read/write section. A disc may also contain a Media Key Block in the read-only section. Furthermore, a device capable of writing also stores a valid MKB in non-volatile storage. When writing encrypted content to a disc, the drive always uses the newest available MKB it can find, according to the version number, assuming it contains a valid signature. The potential sources for the MKB are:

1. The MKB stored in the read/write area,
2. The recorder's internal MKB,
3. The read-only MKB,
4. The source MKB, if performing a copy, and
5. An MKB from a remote server, if performing an on-line operation.

Additionally, if the read-only MKB is found to have an invalid signature then the drive aborts the writing process, even if a newer MKB is available. The drive also updates its stored MKB with the newest version encountered when possible.

Recall that in the pre-recorded setting, a device is still capable of decrypting content after it has been revoked, so long as that content was distributed before the revocation. In the recordable setting new blank media will have an updated MKB that revokes the compromised device, which is the MKB that will be used in writing process. Hence, in the recordable setting a device may lose it's ability to decrypt content created prior to having its keys revoked. Furthermore, at the manufacturer's discretion, a recordable device may choose to update the read/write MKB on any content it encounters, potentially rendering previously accessible content unaccessible. It is currently unknown if any consumer devices choose to exercise this option.

4.2 The Binding Nonce and Encrypted Title Keys

As in the pre-recorded setting each piece of content on the disc is encrypted using a Title Key. The Title Keys are stored in the read/write area of the disc alongside the encrypted content in a Title Key File. Each piece of encrypted content may have its own Title Key File or multiple Title Key Files, depending on the application, or it may have a single Title Key File that is used for multiple pieces of content. In general, a Title Key File may contain multiple Title Keys, and the same Title Key will not used for two different pieces of content. When dealing with write-once media, the Title Key File will contain multiple keys that can be used for future data added to the disc.

Recall that in the pre-recorded setting the Title Key is encrypted using a combination of the Volume ID and a key derived from the Device Keys using the information in the MKB. In the recordable setting the Volume ID is replaced with a special value called the Binding Nonce, a random 128-bit value stored in the protected section of the disc. Each piece of content on the disc

is associated with exactly one Binding Nonce, while a single Binding Nonce may correspond to several pieces of content. The decryption process proceeds as in the pre-recorded case, however the appropriate Binding Nonce is used in place of the Volume ID to calculate a key called the *Protected Area Key*. The Title Key is recovered by using the protected area key to decrypt the Encrypted Title Key, which is then XORed with a hash of the Usage Rules. The result of the XOR is the Title Key. Should a malicious user attempt to modify the Usage Rules, the hash value will be different and an incorrect Title Key will be recovered. The content can finally be decrypted using this newly calculated Title Key.

Whenever the MKB for a Title is updated, or if the value of the Binding Nonce changes, the Encrypted Title Key corresponding to that MKB or Binding Nonce must be updated as well. This is a simple process, however this is a critical period during which the old MKB and Encrypted Title Key are erased but the new MKB and Encrypted Title Key have yet to be written. Should a device fail during this critical period the Title Key would be lost, rendering the encrypted content unaccessible. To prevent the loss of the Encrypted Title Key and MKB, the files containing the old MKB and Title Keys are first renamed to pre-specified filenames, followed by the decryption and re-encryption of the Title Key using the new MKB and Binding Nonce. Once the re-encryption process is complete the temporary files containing the old MKB and Title Key File are removed. Should the drive fail while calculating the new keys the process can be restarted using the temporary backups.

4.3 Encrypting and Decrypting Recorded Content

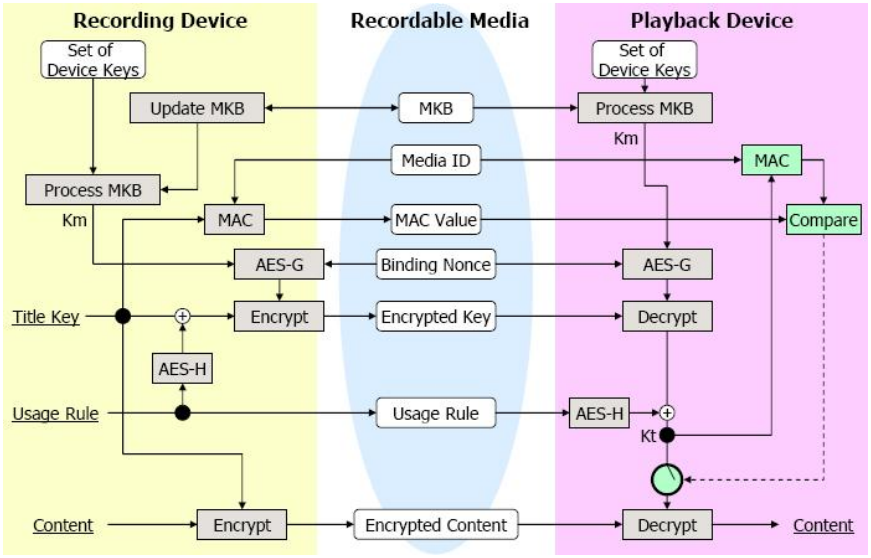

Figure 7: The encryption and decryption process for recordable media. Figure taken from the AACS Recordable Video Book [9].

We now summarize the protocols used to encrypt and decrypt content on recordable media.

Before a piece of content can be decrypted the title key must first be recovered. This is accomplished by first calculating the Protected Area Key:

1. Using the process described in Section 3 calculate the Media Key K_m
2. Using the process described in Section 5.1 request the Binding Nonce B_n
3. Use the AES one way function to calculate K_{pa} from (K_m, B_n)

Once the Protected Area Key is calculated, the content by decrypted:

1. Decrypt the Title Key using K_{pa}, then XOR the result with the AES hash of the Usage Rules to recover the Title Key K_t
2. Using the process described in Section 5.1 request the Media ID M_{ID} and verify CMAC(K_t, M_{ID})
3. Use the K_t to decrypt content

Finally, when new data is to be added to a disc, it is encrypted using the following process:

1. The drive generates a random 128-bit Title Key K_t.
2. Read the Media ID and calculate CMAC(K_t, M_{ID})
3. Encrypt the content using K_t
4. Using the MKB for the content calculate the Protected Area Key K_{pa} and the Encrypted Title Key using AES$(K_{pa}, K_t \oplus$ AES-H(Usage Rules)).

The MKB and Encrypted Title Key can then be written to the disc alongside the content. Figure 7 summarizes the encryption and decryption process.

5 Drive-Host Authentication

When using the AACS in a PC-based system where the Drive and Host are separate entities, both the Host and Drive are issued certificates from the AACS LA. This allows either entity to decide whether or not the other is trustworthy and in compliance with the AACS specifications. These certificates, called the *Drive Certificate* and *Host Certificate*, each contain fields stating the capabilities of device, a unique identifier, the device's public key, and a signature from the AACS LA verifying the integrity of the certificate. A full description of the certificate format can be found in the AACS Introduction and Common Cryptographic Elements specification [7].

Authentication between the Drive and Host occurs each time new media is placed in the drive. This is necessary due to the fact that the new disc may contain updated revocation lists. Recall that along with the necessary information needed to derive the keys to decrypt the content, the MKB also contains entries for the *Host Revocation List* (HRL) and the *Drive Revocation List* (DRL). A Drive may only communicate with a Host that has not been revoked, and a Host may only communicate with a Drive that has not been revoked.

The following is a high level description of the Drive-Host authentication protocol:

1. Drive reads and sends MKB to Host.
2. Both Drive and Host compare their stored revocation lists to the new revocation lists. If the disc contains a newer version, the stored revocation list is updated, making sure that the new revocation list is signed by the AACS LA. The newest version of the revocation will be the one used for the remainder of the protocol.

3. The Drive and Host exchange certificates and random 160-bit nonces.

4. Both sides verify that the other does not appear in the revocation list.

5. If all certificates are valid and neither side has been revoked, the Drive and Host use a key agreement protocol with the Elliptic Curve Digital Signature Scheme using the curve and base-point given in [7].

A full step by step description of the protocol is included as an appendix and follows exactly the description in the specification.

After successfully completing the Drive-Host authentication algorithm, the Drive and Host have derived a shared Bus Key. It is interesting to note that while this key could be used to encrypt messages between the Drive and Host, it is not actually used for this purpose. Instead, the Bus Key is used solely for message authentication by means of including a MAC for any messages between the Drive and Host. The current AACS specifications do not require either the Drive or Host to be capable of encryption and decryption of bus messages, however there is a flag in each certificate stating whether or not an entity is capable of performing bus encryption. The specifications simply state that a description of the bus encryption procedures will be given at a later date.

Another interesting point about the authentication protocol is the process of updating the Host Revocation List and Drive Revocation List if the version on the disc is newer. Recall that even if a player has its Device Keys revoked it is still capable of decrypting any title that has been released prior to the revocation. However, if a Host has its certificate revoked then it can no longer authenticate with any Drive. Thus, the act of participating in the Drive-Host authentication protocol can actually render a software player, or a drive, incapable of authentication even with discs it used to be able to decrypt. This is obviously an undesired consequence for a user, especially a non-malicious user, but vagueness in the specifications may actually work to the user's, or an attacker's advantage.

The specifications do not require the stored version of the HRL or DRL to be updated during authentication, only that the updated version be the one used for authentication. The copy stored in non-volatile storage may be updated at any point before the disc is removed. If the update takes place some time after authentication, this could potentially allow an attacker to terminate the Host or cut power to the Drive before the update takes place, thus preventing a compromised Host (or Drive) from being revoked. While this would not allow the attacker to play any titles with updated revocation lists, it would still be possible to test whether or not a new title revokes your Host or Drive without actually disabling the ability to play older Titles.

5.1 Requesting Media-Specific Information

In the previous section we saw how the Drive and Host established a shared Bus Key. This Bus Key is used to ensure that communications between the Drive and the Host are unaltered. In this section we describe how the Bus Key is used when requesting certain values from the disc, such as the Volume ID or the Binding Nonce. These are necessary when decrypting a disc, as they are used to derive the Title Key, which is in turn used to decrypt the content.

The protocol is simply a direct exchange of messages using MACs to ensure that the message is authentic. Let the message M be one of the following: Volume Identifier, Binding Nonce, Media Identifier, or Pre-recorded Media Serial Number. Figure 8 demonstrates the process in a protocol diagram. When the Host makes a request for M, the Drive reads it from the disc, computes a MAC using the Bus Key, and sends M in plaintext along with the MAC. Upon receiving the M, the host verifies the MAC and decides whether or not the message is valid.

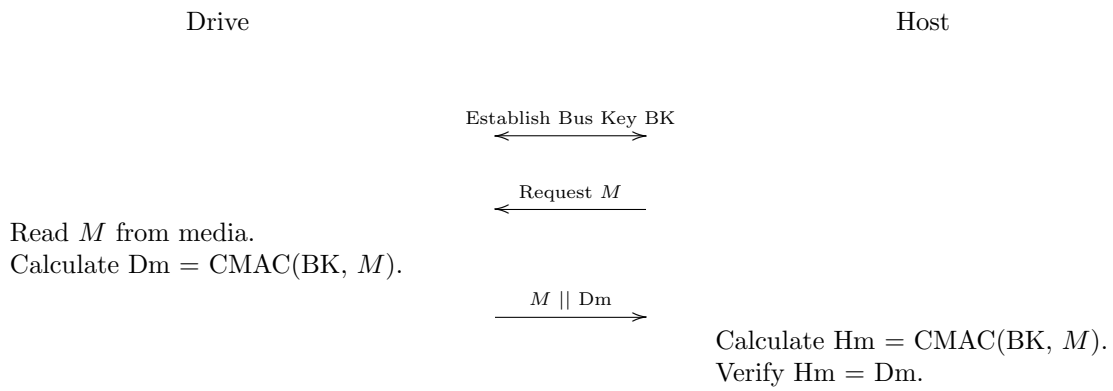

Figure 8: Protocol for transferring media-specific information.

6 On-line Connections

In this section we present the optional enhanced usage of the AACS with on-line connections and also describe how the current attack from Section 3.1 on the basic AACS can be applied to the on-line scenario.

6.1 Defining "Title"

Before looking into the on-line usage of the AACS, we must first more precisely define the word "Title". Intuitively, a Title is just the name of a movie, a TV program, a music album, etc. However, in the AACS, a Title is a distinct path through a piece of content. For example, there is a movie which has five pieces: three basic chapters (1 to 3), and two extra chapters (1.5 and 2.5). Title 1 could be the play list consisting of chapters 1, 2, 3, in order. Title 2 could be the play list that includes all the chapters (1, 1.5, 2, 2.5, 3). In general, different Titles may or may not share some of the same elements.

the AACS defines two kinds of Titles:

- Basic Title: A Title that is readily accessible by the set of Device Keys currently embedded in the device using the Encrypted Title Keys present on the media. Regardless of on-line capability, all the AACS devices are able to access and play Basic Titles (e.g. Title 1 in the example above).
- Enhanced Title: A Title that requires permission over a secure on-line connection or an extended function in the player in order to grant access. Usually, the access grant requires an on-line monetary payment. Title 2 from the example above is an example of an Enhanced Title.

The Title Usage File (TUF) is located on the media and specifies which Titles on the media are Basic Titles, and which are Enhanced Titles, along with the URL associated with the Enhanced Title. The URL directs the playback device to the remote server that can grant the access to the Enhanced Title.

the AACS defines a Title Set to be a group of Titles that are encrypted with the same Title Key. Therefore, a Title Key mentioned in previous sections is actually a Title Set Key. There can be

multiple Title Sets on the media and the corresponding Title Key File may or may not be located on the media.

6.2 Enhanced Modes

There are four different enhanced modes for using the AACS content with on-line connections:

- **AACS On-line Enabled Content** is content that is pre-recorded the media, or part of the initial download in downloaded content, but only made playable following an on-line transaction.

  As described above, a piece of content consists of one or more Title Sets, each of which consists of one or more Titles. All of the Titles in a Title Set are encrypted with a single Title Key. Suppose a Title Set contains a Basic Title and an Enhanced Title, then a legal player will always require permission from on-line transaction to play the Enhanced Title, even if the player already knows the Title Key. However, under the current attack on the basic AACS (3.1), this on-line enabling step can be completely bypassed. Once an attacker possesses the Title Key (through exploiting flaws in software) we can not only play the Basic Title from which the Title Key is extracted, but also play all Enhanced Titles in the Title Set without the required permission. Therefore, the current attack on the AACS can become more serious in the on-line setting.

- **AACS Network Download Content:** An the AACS device with on-line capabilities can download media content from a Remote Server, however, the on-line content cannot be freely distributed after downloading. An on-line transaction protects the content by through *Media Binding*. Simply stated, binding is achieved by using different ways to "protect" the Title Key. In the case of pre-recored media, "protect" simply means "encrypt", and in the case of recordable media, "protect" means to generate a MAC using the Media ID. In total, there are four defined binding methods:

  1. Media Binding: Use the media unique information (Volume ID and Pre-recorded Media Serial Number on pre-recorded media, or Media ID and Binding Nonce on recordable media) to protect the Title Key. Therefore, only with that particular piece of media is the downloaded content allowed to be decrypted.
  2. Content Binding: Use the Title Key already present on the media to protect the content. In this setting, any copy of a particular piece of content can be replayed. For example, if extra scenes for a movie are downloaded using content binding, then anyone possessing a copy of the same movie can replay the downloaded content.
  3. Device/Content Binding: The device independently generates a nonce which cannot be modified by the user. The Remote Server combines this nonce and the original Media Key to create a "Device-and-Content" unique key using a one way function, then encrypts the Title Key using the Device-and-Content unique key. Therefore, only with that particular device, but with any copy of that particular content, can the downloaded content can be decrypted.
  4. Device/Media Binding: This is the most restrictive form of binding. The Title Key is encrypted using a combination of the Device-and-Content unique key and the media unique information. Therefore, only with a specific device and a specific piece of media the downloaded content can be replayed.

- **Streaming Content** is stream content logically associated with pre-recored or network download content, but delivered on demand across the Internet. If a user wishes to replay the streamed content, then permission must be acquired from a remote server.
- **Managed Copy:** Unlike DVDs, where all discs are copy protected by CSS, the AACS allows at least one copy of a Title on pre-recorded media to be copied onto alternative media, such as a home media server or the AACS recordable media. However, the device performing the managed copy must obtain authorization from a remote server, and the appropriate binding method must be used.

Although the AACS specification offers many on-line features, to date no player makes use of on-line connections. Through future updates, the Sony Playstation 3, a network capable device with a Blu-Ray drive, the Microsoft XBox 360, a network capable device with HD-DVD add-on, and PC-based players will likely be among the first to implement on-line connections.

7 Future Work

The AACS provides a traitor tracing mechanism which allows the AACS LA to trace the source of unauthorized copies of protected content. In other words, the traitor tracing scheme in the AACS enables the AACS LA to determine which playback device has had its Device Keys compromised so that they may be revoked. However, it is generally believed that this feature has yet to be implemented for HD-DVD and Blu-Ray discs.

We can demonstrate the basic idea of the traitor tracing scheme used by the AACS in the following example. For an arbitrarily chosen HD-DVD or Blu-Ray movie, select 15 five-second snippets of video and duplicate each snippet into 16 identical copies. Next, apply different fingerprints to each of the 16 copies for all snippets and place each of the fingerprinted copies of each of the snippets onto the disc. By using short lengths of video, this procedure should not induce large storage overhead. Each player is given a special set of keys (Sequence Keys) which allows it to play exactly one of the 16 fingerprinted copies for each snippet. So, for each player (or group of players), we can label a movie with different codewords, $(x_1, x_2, ..., x_{15})$, where x_i represents one of the 16 differently fingerprinted copies of the i-th snippet.

Suppose that the movie King Kong has codeword $(3, 15, \ldots, 7)$ for player p. This means that p can play the 3rd copy of the 1st snippet, the 15th copy of the 2nd snippet, and so on. These codewords are termed the "inner code" and are constructed using Reed-Solomon codes [18]. The inner code is a (15, 256, 16) code, meaning that each codeword in the code is of length 15, and the element in a codeword is from an alphabet of size 16. There are 256 different codewords for a movie. It also implies that the code is 3-TA.

As the name "inner code" may lead one to guess, there is also an outer code. Each codeword in the outer code corresponds to a sequence of movies that a player can play. It has length 255 with each element in the codeword representing a single movie. We can think of each element an integer between 1 and 256, representing one of the 256 versions of the movie (the inner code produces 256 different codewords for a movie, hence 256 different versions). For example, if the codeword for player p is (210, 33, ..., 129), it means that p can play the 210th version of the first movie, the 33rd version of the second movie, and so on. The outer code is also constructed with the Reed-Solomon code (a 9-TA $(255, 256^4, 256)$ code). Each player is assigned a codeword, and therefore the outer code can support up to 256^4 players. Since an element in the outer code represents a version of a movie, we can use the corresponding inner codeword for that movie to denote this element. Hence,

the outer code can be augmented into

$$((x_{1_1}, x_{2_1}, \ldots, x_{15_1}), (x_{1_2}, x_{2_2}, \ldots, x_{15_2}), \ldots, (x_{1_{256}}, x_{2_{256}}, \ldots, x_{15_{256}})),$$

which is referred to as a "supercode".

The implementation of traitor tracing in the AACS is done through a set of Sequence Keys, however, the AACS specification provides very little information on the traitor tracing scheme and very limited explanation of how the Sequence Keys are derived. A handful of papers on the traitor tracing scheme in the AACS, written by its designers, provides a high level overview of the scheme [12], [11], [13], [14], [10]. These papers are the source from which we have learned about the traitor tracing scheme, however, we have not yet be able to find any detailed description of the scheme and its implementation. Unfortunately, one of the papers claims that the details are confidential. It is future work to study and analyze the traitor tracing scheme used in the AACS in more detail in order to determine whether or not it is effective and efficient for such a large system.

8 Concluding Remarks

In this paper we have provided a detailed explanation of the underlying broadcast encryption scheme used by the AACS, and how it is applied for encrypting and decrypting content on pre-recorded media. We also provided a summary of the AACS capabilities for recordable media and on-line connections. Unfortunately, specific details about the implementation of traitor tracing in the AACS is being kept secret, making a proper description of the method difficult.

The AACS makes for an interesting case study, as it is one of the only large scale real-world implementations of a broadcast encryption scheme, and it highlights many of the problems encountered when applying theoretical ideas in a real-world setting. Although the underlying cryptographic primitives and protocols are believed to be secure, implementing them securely remains a challenge. Now that the first set of Device Keys have been revoked, it remains to be seen whether or not the AACS can actually succeed in protecting future content.

References

[1] 4C Entity. "Content Protection for Recordable Media", `http://www.4centity.com/tech/cprm/`, Accessed April 12, 2007.

[2] DVD CCA. "Content Scramble System", `http://www.dvdcca.org/css/`, Accessed April 12, 2007.

[3] Intel et al. "Advanced Access Content System (AACS) - Blu-ray Disc Pre-recorded Book", Revision 0.912, July 27, 2006, `http://www.AACSla.com/specifications/AACS_Spec_BD_Prerecorded_0.912.pdf`

[4] Intel et al. "Advanced Access Content System (AACS) - Blu-ray Disc Recordable Book", Revision 0.92, July 24, 2006, `http://www.AACSla.com/specifications/AACS_Spec_BD_Recordable_0.92.pdf`

[5] Intel et al. "Advanced Access Content System (AACS) - HD-DVD and DVD Pre-recorded Book", Revision 0.912, August 15, 2006, `http://www.AACSla.com/specifications/AACS_Spec_HD_DVD_and_DVD_Prerecorded_0_912.pdf`

[6] Intel et al. "Advanced Access Content System (AACS) - HD-DVD Recordable Book". Revision 0.912, July 25, 2006, http://www.AACSla.com/specifications/AACS_Spec_HD_DVD_Recordable_0.921_20060725.pdf

[7] Intel et al. "Advanced Access Content System (AACS) - Introduction and Common Cryptographic Elements" Revision 0.91, February 17, 2006, http://www.AACSla.com/specifications/specs091/AACS_Spec_Common_0.91.pdf

[8] Intel et al. "Advanced Access Content System (AACS) - Pre-recorded Video Book", Revision 0.91, February 17, 2006, http://www.AACSla.com/specifications/specs091/AACS_Spec_Prerecorded_0.91.pdf

[9] Intel et al. "Advanced Access Content System (AACS) - Recordable Video Book", Revision 0.91, February 17, 2006, http://www.AACSla.com/specifications/specs091/AACS_Spec_Recordable_0.91.pdf

[10] H. Jin, J. Lotspiech, and M. Blaum. "Efficient Traitor Tracing". *International Symposium on Communication Theory and Applications*, 2003.

[11] H. Jin and J. Lotspiech. "Traitor Tracing for Prerecorded and Recordable Media". *ACM DRM Workshop*, 83-90, Washington, D.C., 2004.

[12] H. Jin and J. Lotspiech. "Attacks and Forensic Analysis for Multimedia Content Protection". *ICME*, Amsterdam, Netherlands, 2005.

[13] H. Jin and J. Lotspiech. "Hybrid Traitor Tracing". *IEEE, International Conference on Multimedia and Expo*, 2006.

[14] H. Jin and J. Lotspiech. "Practical Forensic Analysis in Advanced Access Content System". *Information Security Practice and Experience, ISPEC 2006*, Hangzhou, China, 2006.

[15] J. Lotspiech. "Broadcast Encryption". *Multimedia Security Technologies for Digital Rights Management*, Academic Press, Chapter 12, 2006.

[16] T. Martin. "A Set Theoretic Approach to Broadcast Encryption". *Ph.D Thesis*, Royal Holloway University of London, 2005.

[17] D. Naor, M. Naor, and J. Lotspiech. "Revocation and Tracing Schemes for Stateless Receivers". *Advances in Cryptology - CRYPTO '01*, LNCS, volume 2139:41-62, 2001.

[18] J. N. Staddon, D. R. Stinson, and R. Wei. "Combinatorial Properties of Frameproof and Traceability Codes". *IEEE Transaction on Information Theory*, 47:1042-1049, 2001.

A Drive-Host Authentication (Detailed)

The following is a detailed description of the AACS Drive-Host Authentication Protocol. The numbers on each line correspond to the number given to that step in the AACS Introduction and Commond Cryptographic Elements specification. The AGID sent in step 5 is the Authentication Grant Identifier, which is used to identify a specific session in the case a drive can support connections from multiple hosts.

	Drive		Host
1.		$\xrightarrow{MKB}$	Compare version of stored DRL to DRL in MKB. Use newest DRL.
2.			If DRL in MKB is newer, verify signature. Abort if signature not valid.
3.	Compare version of stored HRL to HRL in MKB. Use newest HRL.		
4.	If HRL in MKB is newer, verify signature. Abort if signature not valid.		
5.		$\xrightarrow{AGID}$	
6.			Generate 160-bit nonce H_n
7.		$\xleftarrow{H_n \| Hcert}$	
8.	Verify host certificate type and length. Abort on failure.		
9.	Verify signature on host certificate. Abort on failure.		
10.	Check HRL and abort if Host ID is found.		
11.			Request nonce D_n and drive certificate.
12.	Generate 160-bit nonce D_n.		
13.		$\xrightarrow{D_n \| Dcert}$	

14.			Verify drive certificate type and length. Abort on failure.
15.			Verify signate on drive certificate. Abort on failure.
16.			Check DRL and abort if Drive ID is found.
17.			Request a point on the elliptic curve D_v and its signature.
18.	Generate 160-bit random value D_k		
19.	Calculate $D_v = D_k G$ where G is the base point of the elliptic curve.		
20.	Calculate D_{sig} as the signature of $H_n \| D_v$.		
21.		$\xrightarrow{D_v, D_{sig}}$	
22.			Verify D_{sig} and abort on failure.
23.			Generate 160-bit random number H_k.
24.			Calculate $H_v = H_k G$.
25.			Calculate H_{sig} as the signature of $D_n \| H_v$
26.		$\xleftarrow{H_v, H_{sig}}$	
27.	Verify H_{sig} and abort on failure.		
28.	Calculate Bus Key B_k as the 128 least significant bits of x-coord($D_k H_v$)		
29.			Calculate Bus Key B_k as the 128 least significant bits of x-coord($H_v D_k$)