Piracy Protection for Streaming Content in Home Networks

Hongxia Jin, Jeffrey Lotspiech

Piracy Protection for Streaming Content in Home Networks

Hongxia Jin[1] and Jeffrey Lotspiech[2]

[1] IBM Alamein Research Center
San Jose, CA 95120, USA
jin@us.ibm.com
[2] Lotspiech.com LLC
Henderson, Nevada, USA

Abstract. In this paper we study content protection techniques to defend against piracy for streaming content in home networks where multiple digital devices are connected into a peer-based cluster and seamlessly work together. We are particularly interested in the anonymous re-broadcasting attack where pirates re-distribute the per-content encrypting key or the decrypted plain content. In literature, to defend against an anonymous attack, content is usually built with different variations. For example, content is divided into multiple segments, each segment comes with multiple variations (e.g., watermarks), and each variation is differently encrypted. Each device only has the key to decrypt and play back one variation per segment through the content. The re-distributed keys can be linked back and used to identify the original devices (terms as *traitors*) who were given those keys and involved in the piracy.
This technology works well for prerecorded content scenarios in which a trusted party outside the device pool can deliberately author the content with multiple variations. However it cannot be applied to a peer-based home network when the streaming content is brought into the home network via a peer device who is not a special trust party and who is not allowed to know the secret keys of other peer devices. On the other hand, the trend of the consumer appetite for digital content is increasingly switching from physical media to streaming and internet consumption. In this paper we have designed the first content protection system that allows a recording device inside the home network to bring the streaming content into the home network in a secure way that devices and only devices in the same home network can playback the recording. More importantly, the recorded content without variations can still be used to obtain forensic information, when anonymous piracy attacks occurs, to identify the source devices that participated in the piracy attack. The identified traitorous devices can be revoked for future content access. The technology described in this paper is used to enable the secure sharing of premium quality High Definition content across a consumer's all audio-video devices at its home networks.

1 Introduction

Home networks become more and more popular where people connect multiple digital devices together at home. For example, people have multiple audio-video recording and play-back devices in different rooms and connect them into a cluster (or a home entertainment network). The goal of a home entertainment network is to enable people to access their content anytime anywhere. For example, one device may do the recording in one room, another device may play back the recording in another room. To facilitate this, industry standard such as High-Definition Audio-Video Network

Alliance (HANA) [1] was formed to provide consumers with a simple way to connect and enjoy High Definition (HD) entertainment anywhere. While consumers want to enjoy the convenience brought by a home network, one of the biggest concern is the security of the content. If it is not secure, the content owners will not allow the copy-righted movies to be placed onto people's home network. Similarly, software providers will not allow their copy-righted software to be downloaded into people's home computer networks. On the other hand, streaming becomes increasingly a way to bring content into a home network. It has the obvious convenience that people do not have to leave their home in order to obtain the content. As one can imagine, the content may be encrypted and the digital devices posses secret device keys that enable them to decrypt and play back the content.

1.1 Piracy threat for streaming content in home network

An anonymous attack in the home network scenario is as follows. The attackers compromise one or more sets of device secret keys and set up a server with those keys. They also sell a client with the following value proposition: the client will rip any content in your home network cluster for you, allowing you to make unauthorized copies of purchased content for your friends (or more seriously sell them to anybody on the Internet) or to rip rental content so you can permanently add it to your library after only paying the rental fee.

When evidences of this type of illegal sharing (piracy) are recovered, it is highly desirable to design a content protection system that can detect those compromised device secret keys used in the above pirate server and revoke them for future content access. Such a trace-and-revoke system is what this paper is concerned with. In literature, forensic technology used to detect piracy is generally termed as "traitor tracing". The original devices whose keys are compromised and used in the piracy is called "traitors".

1.2 Existing traitor tracing schemes for physical media

"Traitor tracing" for prepared content, (eg, physical media) has been extensively studied. In order to trace traitors, different versions of the content need to be prepared before distributing to devices. Content is divided into multiple segments and some segments are chosen to have multiple differently watermarked and encrypted variations. The different encrypting keys allow different devices to access content through different pathes. Each different path becomes one content version. The recovered pirated variation encrypting keys or the content version can link back to the actual traitorous devices who were assigned those keys or content versions. Various traitor tracing schemes [3] [8] [9] for anonymous attacks have been designed. Advanced Access Content System (AACS) [5] deployed the "*Sequence Keys*" traitor tracing scheme for prerecorded Blu-Ray DVD disc. All these schemes followed the same paradigm above.

A license agency is responsible for assigning tracing keys to devices and also responsible for detecting traitors using the assigned and recovered tracing keys. For AACS type of system for prerecorded content, the content is prepared with multiple variations by a trusted entity outside the device pool before distributing to all devices/users. This entity has knowledge of every tracing key that was assigned to devices in the system, so it is possible to encrypt the content variations in such a way that every device in the system can only decrypt one variation of the content.

However, one cannot directly apply such a scheme for streaming content. There does not exist a special trusted party (license agency) inside the network cluster. Every

device is a peer. As a peer device, the recording device, who records and creates multiple variations for the content, is not allowed to know the secret tracing keys owned by other peer devices. Furthermore, there does not exist an online license agency outside the network cluster that the recording device can contact and do authorization when recording occurs. Therefore it cannot prepare the multiple variations of the recording content in the same way as the license agency in the prerecorded content scenarios and still guarantee that every other devices in the same network can playback the recording.

The main contribution of this paper is that we will present the first content protection system that allows one peer device (not a central trusted party) to make a recording of streaming content in a secure way that all other peer devices in the network can play back the recording. Furthermore, the recorded content without pre-streaming-prepared variations can still be used to obtain useful forensic information against the above anonymous key-redistribution attack. The detected traitorous tracing keys can be revoked/disabled from accessing future content. Our scheme achieves the same traceability as the counter part of the prerecorded content. Our scheme can be used in HANA [1] to provide secure sharing of premium high definition content across audio and video devices in a consumer's home network.

In rest of the paper, in Section 2, we will introduce our "recording keys", "recording key table", and "title key blocks" concepts to use in our content protection system. Then in Section 3 we will present the overall architecture of our content protection system for peer-based home networks. In Section 4 we will show how to assign recording keys in our system and then in Section 5, we will present our nested traitor tracing scheme using recording keys. In Section 6, we will discuss revocation of detected guilty devices, analyze tracing efficiency and also present some optimizations to improve feasibility during implementation. We conclude in Section 7 for future work.

2 Preliminaries

In our system, we assign a sequence number $1...n$ to each piece of content. When every sequence number is used, it repeats itself from beginning again.

1. Tracing Key K_s
   Each device is assigned a set of tracing keys. Each tracing key corresponds to one content sequence number. For example, for a device, tracing key K_{si} corresponds to its tracing key for content sequence #i. The tracing key assignment is from a matrix with each cell being a randomly generated tracing key. The cells in one column represents all the different tracing keys for one content sequence (eg, movie number one) in the sequence of content.
   Each device gets assigned exactly one tracing key from each column (eg., for each movie). In other words, each device only knows one tracing key for each content sequence number.
2. Title Key K_t
   The title keys are randomly generated for each content and are the actual keys used to encrypt the content. The entire content may be encrypted by one title key; or the content can be divided into multiple segments, and each segment encrypted by a different title key.

3. Recording Key K_r
   Recording keys are used to encrypt the title keys for the content. Recording keys can be static to a cluster of devices in one home network and be same for all

content brought into the home network; or more preferably dynamically created for each streaming content brought into the home network.

If it is dynamic and specific to each streaming content, the recording key table comes into the home network together with the content during streaming, for example, as a header of the streaming content. The recording keys are brought into the home network in the encrypted format in a recording key table.

4. Recording Key Table

   An encrypted recording key table comes with a piece of streaming content. Suppose that streaming content is assigned to be sequence number #i, in each row of the recording key table, it contains a set of recording keys encrypted by the tracing keys for content sequence number #i. The device can use its assigned tracing key for content sequence number #i to decrypt one row of the Recording Key Table and obtain its set of recording keys for this piece of streaming content.

   Figure 1 shows a sample recording key table for streaming movie #i. Each recording key in row #j is actually encrypted by K_{sij}. A device possessing tracing key Ks_{ij} can go to row #j of the recording key table and decrypt to get its set of recording keys, namely Kr_{j1}, Kr_{j2}...

Ksi1	E(Kr11)Ksi1	E(Kr12)Ksi1	...
Ksi2	E(Kr21)Ksi2	E(Kr22)Ksi2	...
.			
.			
.			
Ksij	E(Krj1)Ksij	E(Krj2)Ksij	...

Fig. 1. Encrypted Recording Key Table for movie #i

The name of "recording key" comes from the fact that for streaming content, it is required that one device makes a recording and all other legitimate devices in the same home network cluster should be able to playback the recording. For this reason, the recording keys must have the property that any two devices must share at least one recording key in common. We will discuss more details on recording key assignment in Section 4.

5. Title Key Blocks TKB

   TKB is created by the recording device in the home network when making the recording. For each title key K_t used to encrypt the streaming content, the recording device encrypts that title key with every recording keys it obtains from the recording key table for this content. Each such encrypted title key is an entry of the title key block.

   Figure 2 shows a sample TKB for one title key K_t. It is possible to use multiple title keys to encrypt the content. As one can imagine, different parts of the content may be encrypted using different title keys. For example, a video recording could change the title key every minute of video. In that case, each title key will be encrypted by all the recording keys assigned to the recording device for this content, resulting in multiple TKBs, each TKB looks like the one shown in Figure 2. The recording device can insert the title key blocks as a header for the recording content.

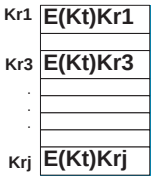

Fig. 2. Title Key Block for a title key K_t used in a recorded movie

3 Content protection system for streaming content

Figure 3 illustrates an overall architecture of a content protection system for streaming content.

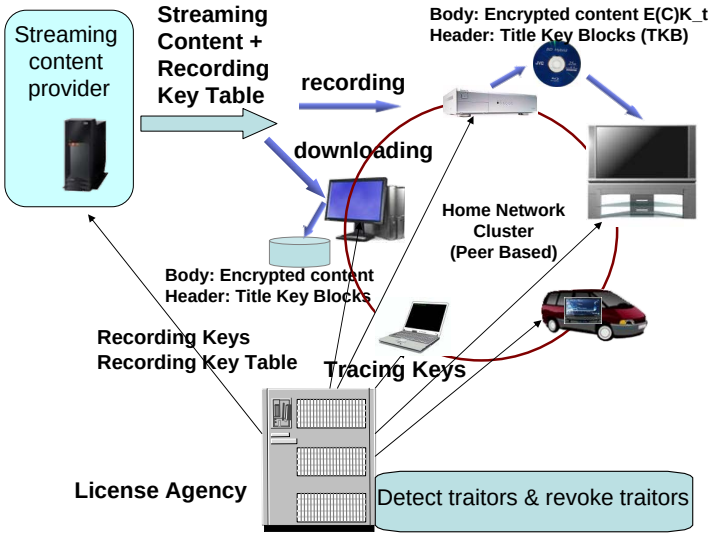

Fig. 3. Architecture for Content Protection system for Streaming Content

In this overall architecture, there are four types of components in the system, the first one is the license agency, similar to that in the prerecorded content case; and the second one is the streaming content provider. The two other components are inside the peer-based home network corresponding to the two types of devices in the home network, one type of devices can record/download the streaming content (for example, a video recorder and a computer); the other type of devices can only playback the recorded content (for example, a high-definition DVD player). Also note that the home network is peer-based. The content streamed by the content provider

does not come with multiple variations, instead it is brought into the home network by a recording device inside the home network. The requirement of a home network is that a recording device can record the streaming content in a secure way that only devices inside the same home network can playback the recorded content. Now we will show how components in this system interplay with each other in this architecture.

3.1 License agency: manage secure content sharing in home network

License agency is a trusted party, it assigns the tracing keys to all devices in a home network cluster. The assignment can be done from a matrix as discussed in last section.

The license agency is responsible for assigning the recording key and creating the recording key table for the content provider. The process takes in the following steps:

1. Assign/create recording keys based on the approach detailed in Section 4.
2. Suppose the current streaming content is sequence number #i, use tracing keys associated with sequence number #i to encrypt the recording keys and create the encrypted recording key table as shown in Figure 1.

We discussed above, in a home network, it requires that one device does the recoding and all other devices in the system can play back the recording. In turn, it requires that recording keys for any piece of streaming content must be created/assigned in such a way that any two sets of the recording keys must share at least one recording key in common. We will discuss the assignment in more details in Section 4.

When piracy occurs, the license agency is responsible for tracing traitors and revoking the guilty tracing keys for future content use. We will discuss these functionalities in Section 5 and Section 6.

3.2 Content Provider: distribute streaming content

As one can imagine, the content provider will stream the content to its users. In addition to the content, it will also provide to the user a recording key table assigned from the license agency for this piece of streaming content. Recall that the recording key table is encrypted by the tracing keys as described in the previous section.

3.3 Recording devices: encrypt streaming content

A recording device in the home network brings the streaming content into the network cluster. It can record/download the streaming content and bind the content into a secure format. For example, a video recorder may record a streaming movie off the air onto a disc; a computer may download a copy-righted software onto a secure file.

Note that the license agency only exist outside the device cluster and only exist offline. For the operations inside the cluster, there does not exist such a trusted party. When recording operation occurs, the device cannot contact license agency to do authorization. In fact, during the recording and binding, the recording device performs the following steps:

1. Use its tracing key to decrypt the recording key table for this streaming content and obtain its set of recording keys $Kr1, Kr2...Krx$.
2. Randomly pick title keys K_t and use them to encrypt the streaming content.
3. Encrypts each title key with all the recording keys it has K_{ri} for this piece of content and create the Title Key Blocks.

As a result, the newly recorded content is encrypted with the title keys and the TKBs become the header of recorded content. In summary, the recorded content package contains the following:

1. The encrypted recording key table coming together with this streaming content
2. The encrypted (i.e., re-bound) content: $E_{K_t}(content)$
3. The encrypted title key block: $\langle E_{k_{r1}}(K_t), E_{k_{r2}}(K_t), \ldots, E_{k_{rx}}(K_t)\rangle$

3.4 Play-back devices: decrypt content

When a playback device in the home network cluster tries to playback the recorded content, the decryption process consists of the following steps:

1. Use its tracing key to decrypt the recording key table and obtain its set of recording keys for this piece of content
2. Use one of its recording key to decrypt the Title Key Blocks and obtain the title key K_t
3. Use K_t to decrypt the content and play back the content

Because of the recording key assignment properties which guarantees that a device shares at least one recording key with the recording keys known/used by the recording device during the recoding, the playback device can decrypt at least one of the entries in the TKB for each title key and obtain the title key to playback the content.

4 Recording keys assignment

The license agency is responsible for creating/assigning recording keys to a piece of streaming content.

There are various techniques to ensure the above mathematical property for recording key assignment, i.e., two sets of recording keys share at least one key. For example, a randomly-generated code can be filtered to have the desired property. Furthermore, each row can be thought of as a "codeword" and each recording key is thought of as a "symbol". A systematic assignment like a Reed-Solomon code is guaranteed to have the minimum difference. For a specific example, if we need to create 1024 codewords of length 15 (15 recording keys per device) and with 16 different symbols in each column (16 different recording keys in each column), we can use a Reed-Solomon code to create 4096 codewords and any two codewords will differ at at least 13 positions. We can filter those 4096 codewords and get the 1024 codewords among which any two codewords are either one or two symbols (recording keys) in common.

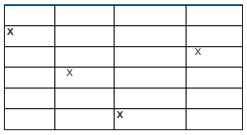

Fig. 4. Recording Key assignment for a movie content

A sample recording key assignment from a matrix is shown in Figure 4. The number of columns corresponds to how many recording keys each device will be assigned. Each row corresponds to how many different recording keys in each column.

It is worth note that while the above mathematical property has to hold in the normal operational case, it does not have to hold in the forensic case. For that reason, in the forensic case it is possible to assign recording keys using codewords that are maximally apart so as to increase the efficiency of the traitor detection. For example, a Maximal-Distance-Separate code like Reed Solomon code using the assignment shown in Figure 4 can serve the forensic situation very well.

It is also possible to design a recording key assignment that is good for both operational and forensic case. For example, one can have K keys spread over C columns. If $C > K/2$, then the code is guaranteed to be overlapping. However, one can also use a larger K if one can do some filtering. An example with 15 keys spreading over 5 columns can generate 317 unique codewords. The most any single key used was 269 times. If we do not need that many codewords, we can filter further. For example, it is possible to obtain 45 unique codewords with the most-used key used only 20 times. This provides a codeword assignment that satisfies the above mathematical property and yet still has very nice spread of the keys. If the license agency uses this kind of recording key assignment for operational and forensic case, the pirate server will have no way to distinguish between the operational and the forensic situation.

5 Traitor tracing using recording keys

The license agency is responsible for identification of traitors (guilty devices that use their tracing keys in piracy) and revoke those guilty tracing keys for future use.

As mentioned in Section 1.1, we are mainly concerned with the following style of attack: the attackers compromise one or more sets of device secret keys and set up a server with those keys. They also sell a client that will rip any protected recording you have made: allowing you to make unauthorized copies of protected recording and sell them. The licensing agency wants to detect which devices and their sets of tracing keys are compromised by the pirate server, and revoke those compromised devices tracing keys for future content access.

Our proposed forensic scheme for streaming media is a nested traitor tracing scheme that consists of two sub-schemes, namely Inner Tracing scheme and Outer Tracing scheme. As shown below, the **Inner Tracing** sub-scheme contains *Inner assignment* and *Inner coalition-detection* tasks. The **Outer Tracing** sub-scheme contains *Outer assignment* and *Outer coalition-detection* tasks.

1. *Outer assignment*: How to assign the recording keys and tracing keys to devices
2. *Inner assignment*: how to assign different title key blocks
3. *Inner coalition-detection*: Repeatedly discover pirated title keys, and trace back to the guilty tracing keys
4. *Outer coalition-detection*: Based on the detected guilty tracing keys from Inner Tracing, trace back to the guilty devices that were assigned those tracing keys

In a forensic situation, the licensing agency buys such a client from the pirate server and carefully crafts some forensic content to probe the pirate client and detect what devices have been compromised and used inside the pirate client.

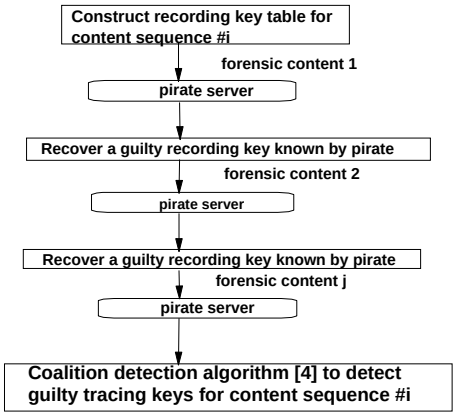

Fig. 5. Inner Tracing based on recording keys to detect guilty tracing keys

5.1 Inner tracing scheme

The goal of Inner tracing scheme is for a particular content sequence number, detect the guilty tracing keys. Recall the tracing keys are assigned from a matrix and each column corresponds to a particular sequence number. Inner tracing scheme needs to detect the compromised tracing key for a particular column.

This Inner Tracing scheme to detect guilty tracing keys is termed as "*forensic analysis based on recording keys*" and is shown in Figure 5. The inner tracing process consists of the following steps.

1. License agency constructs a recording key table for a particular content sequence number #i and encrypts it with those tracing key K_s from column #i of the matrix.
2. (a) License agency builds forensic (recorded) content: pick different title keys K_t to encrypt different content (or different portions of the same content); randomly pick a set of recording key from the above recording key table (for example, from row $\#j$) to encrypt each of the title keys and create the Title key blocks
   (b) Feed the above crafted recorded content to the pirate client for it to decrypt
   (c) The pirate client successfully decrypts the Title Key Block using a recording key it knows and plays back the content
   (d) The content version it plays back reveals which recording key the pirate client has on row $\#j$ to the license agency.
   (e) Iterate the probe by creating another forensic content using a set of recording keys chosen from another row of the above recording key table.
3. After recovering enough number of pirated recording keys, use Traitor Detection algorithm [4] to identify the guilty tracing keys for that column. We will briefly explain the algorithm [4] in Section 5.3.

5.2 Outer tracing scheme

Our scheme has an **Outer Tracing** sub-scheme called *"Forensic analysis based on tracing keys"* as illustrated in Figure 6. This procedure repeatedly invoke the **Inner Tracing** sub-scheme to detect guilty tracing keys in different content sequence numbers (i.e., columns from the tracing key assignment matrix). When enough number of

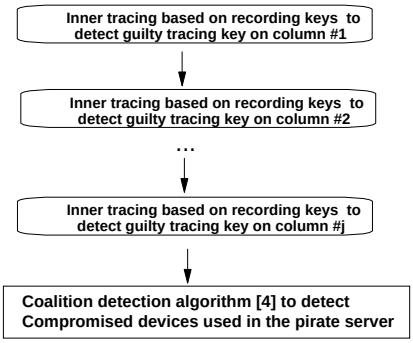

Fig. 6. Outer Tracing based on tracing keys to detect guilty devices

guilty tracing keys are recovered from different columns, the license agency can again use the coalition detection approach shown in [4] to detect the guilty devices.

5.3 Coalition Detection algorithm [4]

In this section, we will briefly explain the coalition detection algorithm that we use in our Inner tracing and Outer tracing scheme. It appeared in [4].

The traitor detection task is the second step of a typical traitor tracing scheme. The first step is how to assign the different keys (or content versions) to different devices. For example, it can be assigned from a matrix, each device gets assigned exactly one key from each column. For example, if a device is assigned a key K_{ij}, it means, for content sequence number #j, this device is assigned key #i for that sequence number. In other words, for each content sequence number, there are multiple (versions of) keys, each device gets exactly one key/version.

When different keys (or content versions) are recovered from piracy on different sequence numbers (i.e., columns), the traitor detection task is to link the recovered keys/versions back to those devices who were originally assigned those keys/versions. The traitor detection is made difficult because of the possibility of attacker collusion. For example, for content sequence number #1, the attackers can redistribute using the version available to colluder #i; when pirating for content sequence #2, the attackers can use and redistribute the version available to colluder #j.

In a typical traitor detection approach, the recovered keys/versions are matched against the keys/versions assigned to the devices. The detection approach will incriminate the devices with the highest number of matchings (sometimes called scores).

In the coalition detection algorithm [4], instead of scoring each individual device, authors proposed to score a set of devices. Their algorithm is based on one important observation: an individual device may score high due to chance alone, while it is much less likely that a coalition (set) of high scoring devices would match all recovered pirate keys/versions due to chance alone.

So the main aim in their algorithm is to find the minimum sized coalition that can explain/match all recovered keys/versions. It is basically the classic Set-Cover algorithm. Given an attack, the tracing algorithm runs by first searching for a coalition of size 1, i.e., a single device. If no such device is found, then the algorithm searches for coalitions of size 2 that explain the attack. If none are found, it searches for coalitions

of size 3 that explain it. Eventually, a coalition will be found as long as the number of recovered keys/versions are enough. The authors claimed several advantages of using their algorithm. One of the main advantage is the efficiency measured by the number of recovered keys/versions needed in order to determine the guilty devices with high confidence. Their algorithm achieves almost linear traceability, meaning, for T attackers involved in a collusion attack, their algorithm takes $O(T)$ number of tests/probes (recovered keys/versions) to identify traitors in that coalition. Due to its superior traceability we employ this algorithm in our traitor tracing scheme.

6 Discussion

6.1 Traceability analysis

As with all other traitor tracing schemes, the traceability of our traitor tracing scheme is measured by the number of forensic testings it takes to probe the pirates and recover forensic feedbacks. In both sub-schemes, we employ the algorithm shown in [4] for the coalition detection task. It takes $O(t)$ number of times to recover forensic feedbacks from pirates when there are t pirates collude in the attack. As a result, the *"forensic analysis based on tracing keys"* needs to be invoked $O(t)$ times, each invocation in turn needs to invoke *"forensic analysis based on recording keys"* $O(t)$ times. Therefore, the total number of probes it takes to detect traitors in a coalition of size t in anonymous attack is $O(t^2)$.

6.2 Revocation after traitor detection

As mentioned earlier, the license agency is responsible for assigning the tracing keys to devices; assigning recording keys and creating recording key tables to give to content providers; tracing traitorous devices involved in piracy; and revoking traitorous devices for future content access.

In our system, assigning tracing keys to devices in the cluster is static. The recording keys and recording key tables are dynamic and specific to each content. In order to disable/revoke those identified traitorous devices and their compromised tracing keys for future content access, the license agency will generate garbage entries associated with those tracing keys when it creates a recording key table for future content. When content provider streams content together with such a recording key table, every guilty device using their tracing keys can only decrypt some garbage (invalid recording keys) out of the recording key tables. Those invalid recording keys will disable the guilty devices from accessing the new content coming together with those recording key tables. That is how revocation works in our system.

6.3 Implementation considerations

From implementation point of view, there are some optimizations we can do to improve the feasibility of our scheme.

We believe it is possible to improve processing efficiency. For example, one can label or name each recording key entry in the recording key table. These labels are encrypted in the table just like the keys. Then, when a device records a header by encrypting the title key with each of its recording keys, it also stores the labels of those keys in the clear. Then, another device can process the header more rapidly, by searching for labels for which it knows the recording key. Otherwise, the device would

have to perform all the decryptions and determine which decryption yields the valid title key by some other means.

We also believe it is possible to reduce the size of the recording key table without reducing security. For example, rather than using full-length keys in the recording table, it is possible to store smaller-length values, and then cryptographically combine those values with the media key to produce a full-length key. For example, if the stored recording keys were 64-bits, and the media key was 128-bits, and the system was using 128-bit AES encryption, the device could expand the 64-bits by concatenating a 64-bit constant defined by the system, and then use the media key to decrypt the resulting value. This "decryption" would be suitably random, and could serve as the actual 128-bit recording key. Thus, the size of the recording key table can be reduced.

7 Conclusion

In this paper we are concerned with piracy protection for the anonymous attack for streaming content in home networks. Unlike traitor tracing schemes for physical media where there exists a trusted party knowing all device secret keys in the system, in a peer-based home network with streaming content, the recording device does not know other peer devices' secret keys, and therefore cannot create the content variations in the recording so that other peer devices can playback the recording later.

In this paper we presented the first content protection system that does not require building multiple variations before the content is streamed. We introduced the concept of "recording keys" that allows one recording device in the network to make a recording in a secure way that all other devices in the network can playback. We proposed a traitor tracing scheme that makes use of recording keys to gain useful forensic information. Our nested tracing scheme takes $O(t^2)$ probes to the pirate server in order to detect guilty pirates of coalition size t.

As future work, we would like to continue improving the feasibility of the scheme. While this itself is not a concern at the cryptographic traitor tracing scheme level, we would like to improve the practicality of our scheme by combining other approaches including signal processing level technologies.

References

1. http://www.hanaalliance.org/
2. http://www.4centity.com/
3. H. Jin, J. Lotspiech and S. Nusser, Traitor tracing for prerecordable and recordable media *ACM DRM workshop, Washington D.C*, 2004.
4. H.Jin, J. Lotspich and N. Meggido, Efficient Coalition Detection in Traitor Tracing *In proceeding of IFIP International conference on Information Security 2008, Sept 8-10, 2008, Milan, Italy*
5. http://www.aacsla.com/specifications
6. B. Chor, F. A, and M. Naor. Tracing traitors. In *Crypto'94, Lecture Notes in computer science, Springer-Verlag, Berlin, Heidelberg, New York*, volume 839, 1994.
7. D. Naor, M. Naor and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", *Crypto 2001, Lecture Notes in computer science*, Vol. 2139, pp 41-62, 2001.
8. A. Fiat and T. Tassa. Dynamic traitor tracing. In *Crypto'99, Lecture Notes in computer science, Springer-Verlag, Berlin, Heidelberg, New York*, volume 1666, 1999.
9. R. Safani-Naini and Y. Wang, "Sequential Traitor tracing," *IEEE Transactions on Information Theory*, 49, 2003.