HYBRID TRAITOR TRACING

Hongxia Jin and Jeffery Lotspiech

IBM Almaden Research Center
San Jose, CA, 95120

ABSTRACT

In this paper we study the traitor tracing problem, a technique to help combat piracy of copyrighted materials. When a pirated copy of the material is observed, a traitor tracing scheme allows one to identify at least one of the real users (traitors) who participate in the construction of the pirated copy. The authors have been involved in what we believe is the first large-scale deployment of the tracing traitors approach. In this paper we shall present a key management scheme that provides the system designer flexibility, in that he/she can choose to trace down to an individual user or only trace to a device's model. It helps effectively defend against class attack and evil manufacturer problem. It also provides a technology to protect consumers' privacy when needed. This flexibility is believed to be needed and has been adopted by content protection standard AACS for next generation DVDs.

Keywords: key management, traitor tracing, watermarking, next generation DVD

1. INTRODUCTION

This paper is concerned with the protection of copyrighted materials in a one-to-many distribution channel. Some business scenarios in this category concern that the copyrighted content is only distributed to paying customers. For example, a pay-TV broadcast system or selling copyrighted music content through the Web. Another type of business scenario in this category, and the one we have been working on directly, is massively distributing prerecorded media. The success of these business models hinges on preventing the use of unauthorized copies of the media.

In July 2004, eight companies, Disney, IBM, Intel, Matsushita, Microsoft, Sony, Toshiba, and Warner Bros., announced that they had come together to work on content protection for the new generation of high-definition DVD optical discs. The technology they are developing is called *Advanced Access Content System*, or AACS [1]. The fundamental protection of the AACS system is based on broadcast encryption with a subset-difference tree using device keys and a media key block. It allows unlimited, precise revocation without danger of collateral damage to innocent devices. The mechanism is designed to exclude clones or compromised devices, such as the infamous "DeCSS" application used for copying "protected" DVD Video disks. Once the attacker has been detected, newly released content incorporates new media key blocks which exclude the keys known to the attackers. To identify which keys have been used in the compromised devices, a forensic media key block, a type of carefully crafted test media key block, is fed into the device. The observed results allow determination of the keys used in the device.

However, the AACS founders do not believe that this level of renewability solves the piracy problem completely. In fact, because of the inherent power of the revocation of the AACS system, it is possible that the attackers may forgo building clones or non-compliant devices and instead devote themselves to server-based attacks where they try to hide the underlying compromised device(s). This is progress, because these server attacks are inherently more expensive for the attackers. However, AACS found it desirable to be able to respond to even these types of attacks. In one particular attack, you could imagine the attackers building a server that distributes per-movie keys. Of course, the attackers would have to compromise the tamper-resistance of one or more players to extract these keys. With a compromised player, it is also possible to get an exact in-the-clear digital copy of the movie, with all of its extra navigation and features, and distribute this. In these cases, the only forensic evidence availability are the per-movie keys or the actual copy of the content. To help defend against these types of attacks, the AACS system adopts tracing traitors technology. The authors have been involved in what we believe is the first large-scale deployment of the tracing traitors approach.

The real user who participates in this piracy is called equivalently either a *traitor* or a *colluder*. A traitor tracing scheme allows to identify at least one of the traitors. In this case the only way to trace traitors is to use different versions of the content/keys for different users. This allows the rebroadcasted decryption keys or contents to be linked to the group of users who were given that version.

Unfortunately in many cases it is infeasible to send a different version of the content to each user. What is possible, however, is to create a limited number of versions. For example, each content is divided into multiple segments, some segments are chosen to have differently marked variations. The content with all the small variations at chosen points in the content is bulk-encrypted and put on the disc distribute to each user. However, each user plays back the content through a different path, which effectively creates a different content version. We have designed a systematic way to create these different versions [2]. However, in general, how those different versions are created is irrelevant to the discussions in this paper. Let us suppose there exists an "inner code" that creates multiple versions of the content to be distributed. To simplify our discussion, we will use movie as a sample content throughout the rest of the paper.

In our model, we assume that each movie comes with multiple versions created from "inner code", for example, 256 versions. There will be 255 movies in the system. Traitors can collude and use one traitor's version for one movie and use another traitor's version for another movie. Unfortunately when a sequence of pirate movies are recovered, the sequence could happen to belong to an innocent user. A weak traitor tracing scheme wants to prevent a group of colluders from thus "framing" an innocent user. A strong traitor tracing scheme allows at least one of the colluders to be identified.

Traitor tracing problem was first defined by Fiat and Naor in a broadcast encryption system [3]. The security threat in that system is that a group of colluders can construct a clone pirate decoder that can decrypt the broadcast content. The threat model that this paper is concerned with is the "anonymous attack". Attackers can construct a pirate copy of the content (content attack) and try to resell the pirate copy over the Internet. Or the attackers reverse-engineer the devices and extract the decryption keys (key attack). They can then set up a server and sell decryption keys on demand, or build a circumvention device and put the decryption keys into the device. For content attack, it requires good watermarking schemes that embeds different information for every variation.

A tracing scheme is static if it pre-determines the assignment of the decryption keys for the decoder or the watermarked variations of the content before the content is broadcast. The traitor tracing schemes in [4, 5] are static and probabilistic. Sequential traitor tracing is presented in [6]. All the above mentioned traitor tracing methods are designed to trace to the individuals.

However, in real applications, sometimes the licensing agency only wants to trace to the manufacturer/model of a particular leak, instead of revoking an individual device. Why is this? Certain attacks are what have been called *class attacks*. A manufacturer may have made a mistake that makes it easy for people to extract the keys from the device. In this case, the key revocation mechanism might be overwhelmed by the sheer volume of the number of compromised keys. A more effective response might be to cooperate with the manufacturer to distribute firmware updates along with the content. Alternatively, code might be distributed with the content that "sniffs"—that is, it checks to see if it is running in the legitimate environment or a circumvention platform, and refuse to play in the latter. Also, in some systems where the tracing is not identify a device but instead an individual, privacy concerns might come into play. In fact, in the case of a prepackaged physical media, one has to consider the question if the licensing agency is even entitled to revoke a player if its owner has used it to make unauthorized copies of the content. The question is still under debate. We believe it is desirable to have a hybrid approach that flexibly allows either tracing to manufacturer/model or to individual device, depending on the different needs at different times. We have designed a practical and systematic key management approach to assign the variation/keys to devices so that it is possible to switch on and off the ability to trace to models or to individuals.

The rest of the paper is organized as follows. We will present a new key assignment scheme that enables the hybrid tracing in Section 3. We will also show how to defend against the evil manufacturer problem in Section 4. We conclude our work in Section 5.

2. BASIC SCHEME

Any static traitor tracing scheme has two basic steps:

1. Assign movie key versions to users.
2. Based on the observed re-broadcasted keys or contents, trace back the traitors.

This paper will mainly focus on the key assignment scheme used in the first step. The second step can use a straightforward brute-force approach. For each user we can simply compute the number of movies that his or her assigned copies matches with the observed pirated copies of movies. The scheme incriminates the user who has the largest matching with the pirated copies of the movies found.

For the first step, the movie version assignment can be done randomly or systematically, for example, based on an error-correcting code. Assume that each movie has q versions and that there are n movies. We represent the assignment of movie versions for each user using a *codeword* $(x_0, x_1, \ldots, x_{n-1})$, where $0 \leq x_i \leq q-1$ for each $0 \leq i \leq n-1$. We call this level of code "outer code". For example, suppose the inner code creates 256 versions for each movie, then the outer code will have q set to be 256. The outer code is assigned from a key matrix. Columns corresponds to movie sequences and each column corresponds to one movie. Rows corresponds to movie key versions. Each column contains many, even thousands, rows. For example, if there are 1024 movie key versions for each movie and there are 255 movies in the entire sequence, then the key matrix has 255 columns and 1024 rows. Each device is assigned a single key from each column.

On the other hand, it needs 256 tables on the disc. One table per movie version. Each table may contain any information needed for one movie version, for example, the actual encrypting keys for the movie can be put in the table. Each table is encrypted by its corresponding movie key version for that movie. For example, for movie #44, the table #i is encrypted by movie key version #i for movie #44.

Each user will be assigned a unique set of keys corresponding to the above outer code. Notice that this assignment has to be done in advance at the device manufacture time before any content can be distributed. However the decision of tracing to individual or model only is on-the-fly at content distribution time and can vary movie-by-movie. The assignment must be done in a way that, once the keys are assigned to the devices, the authority could choose to trace back only manufacturers/models (with low cost) or trace back end-users (with high-cost).

3. HYBRID TRACING

In order to trace to models, one can always trace to devices first, then find out their models. However, a class attack might be so severe that the size of the coalition overwhelms the tracing. If they are all in the same model, however, they are in a coalition of size 1 for that type of tracing. Also, since there are many fewer models than there are individual devices, model tracing is inherently faster. Clearly, it is desirable to have an approach to flexibly allow both types of tracing to work in some attacks, and in other attacks only allow detection of the manufacturer/model and disable the detection of individual traitors.

Note that there are two naive ways to enable both tracings to work. One way is to store 2 sets of 255 keys in the player. One set of 255 keys are used in a scheme for tracing only to manufacturer/model, the other set of 255 keys are used in the scheme for tracing to devices. Of course, the drawback of this is that it doubles the storage requirement, which is a cost for the device manufacturers.

The second naive way is to put in 256*256 tables instead of 256 tables on the disc. We know that each player stores 255 movie keys that can be assigned based on the outer code. However, note that the outer code specified above is based on tracing to individuals. As mentioned above, when tracing down to an individual, the scheme needs 256 tables on the disc (although, as explained in the last section, we actually use more), and each table is encrypted by its corresponding movie key version for that movie. For example, for movie #44, the table #i is encrypted by movie key version #i for movie #44. Remember, nominally, each sequence key comes with 256 versions. When tracing only to manufacturer/model, for any movie, the individuals within the same manufacturer/model should get the same movie version assignment. However, when the sequence keys are assigned on the outer code specified above, they are based on tracing to individuals. As a consequence, most individuals are not assigned the same movie key version for a movie. In order for the players to be able to play back, each table needs to be duplicated 256 times, and encrypt each of these 256 tables using all 256 movie key versions. This results in 256*256 tables per movie (or much more, if you are using the flexible scheme in the previous section). Clearly, the drawback of doing this is the amount of space required on the disc, which is a cost to the content owner/distributor.

In order to have a practical scheme, we must reduce the cost for the storage requirement both in the device and on the disc. We have designed an efficient key assignment that enables one to perform hybrid tracing without incurring any additional storage cost on the device and the disc.

3.1. Key assignment for hybrid tracing

In order to enable hybrid tracing, we introduce a new concept called "slot". Now the rows in the key matrix are grouped into clusters. A slot is defined to be an assignment of row clusters, one cluster for each column. At any given column, two slots are either identical or completely disjoint. Slots can be assigned to individual manufacturers/models and the keys within the clusters are assigned to the devices made by the manufacturer/model. In effect, the outer code is now itself a two-level code. The entire system is now a three-level code.

Figure 1 shows a toy example. The first level codes assign clusters to the manufacturer/models X and Y and the second level codes assign keys to players A,B within model X, and players C, D within model Y. Model X gets the slot (1,3,4,1), which means it is assigned cluster #1 for movie #1, cluster #3 for movie #2, and etc. Note that the second level code is the assignment inside the cluster. For example, player A gets *(1,1,3,3)* within the clusters assigned to model X, which makes its actual key assignment be $(1, 9, 15, 3)$ from the key matrix.

As a more real example, suppose each movie key comes with 512 versions. So the key matrix has 512 rows and 255 columns. We divide all 512 rows into 32 clusters, each cluster contains 16 rows. The first level code is about the cluster assignment throughout the 255 movie sequence. Each codeword is a slot. So the code has q=32 and n=255. Each slot is given to one manufacturer/model. And a given manufacturer/model may get multiple slot assignments if it is

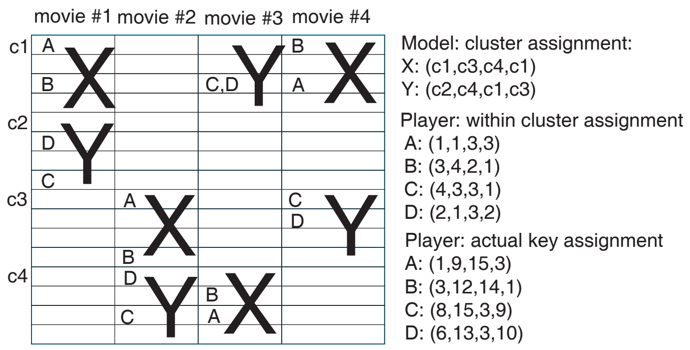

Fig. 1. **Key assignment using "slots"**

producing a lot of devices. The second level code is about the sequence key assignment within a slot. So the code has q=16, and n=255. Each player gets the sequence key assignment based on this code. After the movie keys are assigned based on slots, the movie keys are stored inside the players.

Of course, the choices of the parameters when using the slots also depends on how many manufacturers/models and how many devices the system need to support. Just by way of example, if each movie key comes with 512 versions in this world and there are 512 movie versions created from inner code. Suppose the system needs to accommodate at least 16000 models and at least 1 billion devices. A q=32, k=3, n=255 code for the slot assignment allows to accommodate about 32,000 (precisely 32^3) manufacturer models. And another q=16, k=4, n=255 code can be used to assign movie sequence keys to about 65536 (precisely 16^4) devices within each manufacturer model. The total number of devices the scheme can accommodate is 2 billion. If each movie key comes with 256 versions in this world and there are only 256 movie versions created from the inner code, we can also divide all 256 rows at each column into 16 clusters. In this case we have to use $q = 16, k = 4$ code to assign slots to manufacturers, accommodating 65k manufacturers. The same q=16, k=4 and n=255 code can be used to assign sequence keys to the devices with the model. As a result, the scheme can totally accommodate 4 billion devices.

3.2. Hybrid tracing

In the case of tracing only to manufacturer/model, the capability of tracing to devices can be disabled. This can be done by sending the same movie version to the individuals of the same model. Basically you can choose a movie version from the cluster and its corresponding table can be duplicated to the number of times equal to the number of devices within each cluster. Each duplicated table is encrypted with one of the movie keys within the cluster. In the example shown above, the 512 rows are divided into 32 clusters and there are 16 rows in each cluster. Instead of having 512 different tables as is the case when allowed to trace to devices, there will be only 32 different tables, but each of the 32 tables is duplicated 16 times. All the 16 devices within the same cluster will use their own movie keys to encrypt the 16 identical tables for the cluster. Note that, when doing this, the number of total tables for the movie on the disc is still 512, same as when tracing to devices. But some of the movie versions are not used, only 32 version are used. Of course, when doing this, there is no way for the tracing agency to trace down to devices.After all, every user within the same model get the same movie version.

This scheme allows the detection of the pirate manufacturer/model using less recovered movies than that needed for tracing to devices. The reason of this is not hard to observe. After the sequence key assignment shown above, the outer code used for manufacturer/model tracing would be simply be the first level code used for slot assignment. However, the outer code used for individual tracing would be the combination of the two levels of the assignment shown above. The property of the codes determines that it is more efficient for tracing only to models than tracing to devices. In fact, after the pirate model is detected, with more recovered movies, the individual traitor can also be detected.

Our hybrid tracing scheme allows detection of both collusion between manufacturers/models and collusion between devices. Furthermore, whether the content owner chooses to prepare the movie for the purpose of tracing individual players or of tracing the manufacturer/model is transparent to the players, and can be a movie-by-movie decision. The number of tables does not change based on this decision.

Apparently, with this scheme to do flexible hybrid traitor tracing, it keeps the space requirement small both on the disc and in the player. In fact, the storage requirement on the player and the number of tables on the disc for hybrid tracing are kept same as when the scheme is used only for individual tracing. This is an advantage of our scheme compared to the naive approaches shown in the previous section.

On the traceability side, one strong advantage of our scheme is to flexibly allow switching on and off the capability to trace to devices. Another advantage of using this scheme is that the scheme allows to handle collusion between manufacturers/models as well as collusion between devices.

4. PROBLEM: EVIL MANUFACTURERS

AACS was also motivated by the need to defend against the so-called “evil manufacturer attack” where a licensed manufacturer misuses all the keys assigned to him. When this attack happens, it is desirable to take action against that manufacturer without harming other innocent manufacturers’ devices. Interestingly enough, our key management approach can be highly resilient to the attack that many devices within the same model line are compromised.

There are two general types of attacks. One is the random individual hacking events. The other is the evil or sloppy manufacturer who misuses all the keys assigned to them and cause all those keys be exposed. To a lesser extent, attackers reverse-engineer multiple devices from the same manufacturer/model and compromise many keys assigned to the particular manufacturer/model. In fact, if the sequence keys are assigned randomly from the entire key matrix, an evil manufacturer could quickly learn all the keys in the matrix and break the system. On the other hand, When an evil manufacturer attack occurs, all those keys can be exposed, it can be equivalent to multiple random individual attacks. It turns out that the “slot” idea explained in the previous section is also a very good way to defend against the evil manufacturer problem.

It is interesting that our scheme can defend effectively against both evil manufacturers’ attacks and random individual attacks. Intuitively we want to minimize the overlap between slots. Our systematic assignment of the keys to the devices provides a deterministic guarantees of the Hamming distance, thus the maximum overlap between slots. The Hamming distance can be so big that a collusion up to certain number (for example, m, decided by the Hamming distance) of evil manufacturer models cannot completely cover any given innocent device’s sequence keys. In other words, the probability that a given device’s sequence keys are covered by m manufacturer models are zero. On the other hand, with random assignment, suppose there are q clusters, a given device’s sequence key can be entirely covered by a device in another manufacturer model with probability $(1/q)^n$ where n is the number of sequence keys each device gets. This probability is small, but not zero. Similarly the probabilities that it can be covered by multiple manufacturers are not zero. The systematic assignment is a better option than random assignment.

5. CONCLUSIONS

In this paper, we study the problem of tracing the legitimate users (traitors) who instrument their devices and illegally resell the pirated copies by redistributing the content or the decryption keys on the Internet.

We have presented a key assignment approach that can enable one to flexibly trace to individual devices or only to the models of the devices used in the attack. The advantage of the scheme is that the capability of tracing to the individual device can be switched on and off based on particular application requirement. It allows faster tracing to models to defend against ”class” attack and protects user’s privacy in the case devices can be tied back to users. It also defends well against evil manufacturers. As future work, we would like to continue focusing on overcoming the barriers to bringing the traitor tracing technologies to real practice. We will also focus on detail implementations during its deployment.

6. REFERENCES

[1] www.aacsla.org

[2] H.Jin and J. Lotspiech, ”Attacks and forensic analysis for multimedia content protection”, *ICME*, July, 2005, Amsterdam, Netherlands.

[3] A. Fiat and M. Naor, “Broadcast Encryption,” *Crypto’93, Lecture Notes in computer science*, Vol. 773, pp480-491. Springer-Verlag, Berlin, Heidelberg, New York, 1993.

[4] B. Chor, A, Fiat and M. Naor, “Tracing traitors,” *Crypto’94, Lecture Notes in computer science*, Vol. 839, pp480-491. Springer-Verlag, Berlin, Heidelberg, New York, 1994.

[5] B. Chor, A, Fiat, M. Naor and B. Pinkas, “Tracing traitors,” *IEEE Transactions on Information Theory*, Vol 46(2000), 893-910.

[6] R. Safani-Naini and Y. Wang, “Sequential Traitor tracing,” *IEEE Transactions on Information Theory*, 49, 2003.