Efficient Coalition Detection in Traitor Tracing

Hongxia Jin, Jeffery Lotspiech and Nimrod Megiddo

abstract In this paper we study the traitor tracing problem for re-broadcasting attack. In this attack, instead of building a pirate clone device (or program) based on their secret keys and sell the clone, the attackers want to stay anonymous by redistributing the decrypted content or the actual encrypting keys for the content. To defend against this type of re-broadcasting attack, the content and its actual encrypting key must come with different versions. In our setting, content is divided into multiple segments, each segment comes with multiple variations and each variation is differently encrypted. Each user/device can only play back one variation per segment through the content. A typical traitor tracing scheme for re-broadcasting attack involves two basic steps, assigning the key/variation to devices (assignment step) and detecting at least a traitor in the coalition when a series of pirated key/content are recovered (coalition detection step). We take into considerations of some realities that have been overlooked in existing schemes. As a result, we have designed a probabilistic coalition detection algorithm that is not only closer to real world scenarios but also more efficient than existing approaches. The traceability is defined in terms of the number of recovered pirate copies of the content needed to detect traitor(s) as a function of the number of traitors involved in a coalition. While existing schemes try to identify traitors one by one, our probabilistic algorithm can identify multiple traitors simultaneously and deduce the coalition size during tracing. Therefore, for the same number of total traitors in a coalition, our scheme allows the detection of all the traitors using less number of recovered copies. The superior efficiency of the our coalition detection algorithm made its adoption by AACS (Advanced Access

Hongxia Jin & Nimrod Megiddo
IBM Almaden Research Center
San Jose, CA, 95120
e-mail: {jin,megiddo}@us.ibm.com

Jeffery Lotspiech
Lotspiech.com, Henderson, Nevada
e-mail: jeff@lotspiech.com

Content System) content protection standards for next generation high-definition video optical disc.

1 Introduction

This paper is concerned with the protection of copyrighted materials. A number of business models has emerged whose success hinges on the ability to securely distribute digital content only to paying customers. Examples of these business models include pay-TV systems (Cable companies) or movie rental companies like Netflix, and massively distributing prerecorded and recordable media. These typical content protection applications imply a one-way broadcast nature. A broadcast encryption system [3] enables a broadcaster to encrypt the content so that only a privileged subset of users (devices, set up boxes) can decrypt the content and exclude another subset of users. When a broadcast encryption system is used for content protection, the enabling building block is a structure called Media Key Block (MKB) which is based on hybrid encryption. Each device is assigned a set of unique secret keys called device keys. The media key, which is indirectly used to encrypt the content, is encrypted by device keys again and again and put into MKB which is distributed together with the content. Each compliant device using its device key processes the MKB differently but gets the same correct media key to decrypt the content, while the excluded (revoked) devices cannot decrypt the MKB and get the correct media key.

Note that there can be different pirate attacks in the above content protection system. In one attack, a set of users (device owners) attack their devices, extract device keys out of the devices and use those keys collaboratively build a clone pirate device that can decrypt the content. When a pirate device is found, a traitor tracing scheme enables the broadcaster to find at least one of the users (called traitors) who have donated their device keys into the pirate device. Most existing broadcast encryption and traitor tracing schemes [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] targeted on this type of ”pirate device attack”.

This paper’s focus is on a different attack, namely, the re-broadcasting attack as defined in [15, 16]. When an attacker re-broadcasts the content in the clear, the only forensic evidence is the unprotected copy of the content and the attackers can stay anonymous. The attacker can also simply re-broadcast the media key to stay anonymous and avoid being identified. To defend against the re-broadcasting attack, for different devices, not only the content needs to be in different versions, it also needs to be differently encrypted. Of course sending different versions to different users is oftentimes too costly in bandwidth or disc space needs. To reduce the cost, each content is divided into n segments and each segment is augmented by q different variations which are differently marked and encrypted. The underlying scene remains identical. To save cost, this same augmented content is distributed to every user. However, each user can only decrypt one of the variations at each segment. In other words, each recipient would follow a different path through the variations during playback time. In this way, even though each user does not receive a differently distributed content, it effectively creates different versions of the content for different users.

A traitor tracing scheme in this category usually consists of two basic steps:

1. *assignment step*: Assign a version of the content to the device by assigning the playback path, i.e., which variation to play for each augmented segment in the content.
2. *traitor/coalition detection step*: Based on the recovered pirated content/keys, trace back to the traitors.

The focus of this paper is on the traitor/coalition detection step. In literatures a traitor tracing scheme has been defined as a way to detect at least a traitor in the system when forensic evidence is recovered. Therefore the goal of the traitor detection step, as well as the design of a traitor tracing scheme, like [15, 16, 17, 18, 19], is to identify a traitor. It is assumed that the identified traitor can be disconnected from the system and the tracing continues after that. Indeed, for the *coalition/traitor detection* step existing schemes always use a highest-score approach, where each player is scored based on the number of matching between the recovered pirate movies and the versions assigned to the player, hoping the highest scored player is the guilty traitor. We believe using the one-by-one detection scheme for re-broadcasting attack is inefficient. We are motivated by the fact that in reality the ultimate goal is to detect all traitors in the coalition. If possible, one should try to detect multiple traitors simultaneously rather than detecting one by one. The efficiency of a tracing scheme is measured by the total number of recovered movies it needs in order to detect all traitors in a coalition of size T.

The second motivation of this work has to do with the fact that in reality the coalition size is usually unknown. As a result, the answers a traitor tracing scheme gets are always qualified in real applications. One cannot perform deterministic tracing as those shown in existing work. In reality, tracing will have to be probabilistic. Indeed, the real world question is how to accurately detect traitors without knowing the coalition size and with what probabilities.

We have designed the first traitor/coalition detection algorithm that tried to detect multiple traitors in the coalition together and also deduce the coalition size during tracing. In our algorithm, firstly, with the recovered movies, using set-cover, we try to detect which coalition of players may have involved in the attack, instead of which one particular player may have been involved. Second, when we find a suspect coalition, we cannot trivially incriminate all the players in the suspect coalition. We have designed ways to identify and incriminate the guilty individuals in the suspect coalition. As a result, we could incriminate more than one player at each iteration of the algorithm. Our goal is to correctly identify the actual traitors with high probability. In fact, our algorithm can identify traitors with any confidence the license agency wishes.

The above idea may look simple. But one might have been concerned with the theoretically exponential blow-up in computation time. Fortunately, in reality we find not only computational time is very manageable, we also find the computational time is less an issue than the demand for large number of pirate movies need to be recovered in order to detect traitors. After all, the tracing agency does not have control on how often attackers pirate and re-distribute movies, maybe every week or every month. Therefore, it is much more important to reduce the number of recovered movies that a traitor detection algorithm needs in order to detect traitors. Indeed as pointed out earlier, in this paper we define the efficiency of a traitor detection algorithm to be the number of pirate movies needed in order to detect traitors for a coalition of size T.

Furthermore, the efficiency of our algorithm derives from a very important but maybe counter-intuitive observation, it is much faster to eliminate the completely innocent coalitions than eliminating innocent individuals even though there exist a lot more (i.e., exponential number of)coalitions than individuals. This is because that it is much less likely that coalitions appear by random chance, than that individual players randomly have high scores. This truism is the essence of the efficient tracing underneath our new tracing algorithm.

The authors have been involved in what we believe is the first large-scale commercialization of a tracing traitors system for re-broadcasting attack within the AACS (Advanced Access Content System) content protection standards for next generation of high-definition optical DVDs. AACS adopts the use of the scheme in [20] as the assignment step to satisfy some practical restrictions for the assignment step. But the detection step shown in [20] is not efficient and practical enough. The algorithm we will show in this paper takes into considerations of those realities shown above that have largely been overlooked in existing work. As a probabilistic algorithm, it is not only more practical than deterministic tracing; it is also much more efficient due to the fact that it detects multiple traitors together. Furthermore, the entire tracing can be done in very reasonable time. As a result, AACS adopts the use of the scheme in [20] as the assignment step and adopts the work presented in this paper as the coalition detection step.

In rest of the paper, in Section 2, we will first provide more contextual background for designing a traitor tracing scheme for AACS for re-broadcasting attack, including some practical restrictions on the assignment step. Many schemes shown in literatures do not have those restrictions in mind and thus do not satisfy those restrictions. We will summarize the result in [20] to show how it can satisfy some of AACS' restrictions on assignment step. Then we will show our traitor/coalition detection algorithm in Section 3. We will analyze its false positive rate in Section 4 and its performance/efficiency in Section 5. We show simulation results in Section 6 and conclude in Section 7. For concreteness, we use movie as a sample content in this paper.

2 Background for traitor tracing in AACS

AACS founders find it acceptable to makes the following marking assumption on the pirate model. Given two variants v_1 and v_2 of a segment, the pirate can only use v_1

or v_2, not any other valid variant v_i. In a key-rebroadcasting attack, this assumption says if attackers have two valid random cryptographic keys, it is probable they will simply redistribute them instead of calculating a valid random key from the known two random keys since it is difficult if not impossible to do so. For content re-broadcasting attack, while watermarking is a common way to build variations, there are other better ways to exploit and satisfy the marking assumption. It is outside the scope of this paper to discuss why AACS adopts this attack model.

When a movie is divided into multiple segments and each segment is augmented with multiple (q) variations, as one can imagine, those variations take extra space on the disc. For content owners, a practical traitor tracing scheme on a prerecorded optical movie disc should take no more than 10% of the space on the disc to store the variations. This puts practical restriction on the number of variations one can put into a movie. The market for such discs is huge, involving literally a billion playing devices or more. This means a tracing scheme needs to be able to accommodate large number of devices. While these restrictions are inherently conflicting, a practical traitor tracing scheme must meet these requirements first. After meeting these requirements, it is also important to detect the coalition of traitors using as few recovered movies as possible.

In summary, a traitor tracing scheme for AACS needs to meet all the following requirements:

1. the number of variations for each movie cannot be big
2. the number of devices/users *must* be big
3. after the above two requirements are met, the number of movies necessary to detect a coalition of should be as small as possible

It is very important to notice that much of the literature on traceability codes has taken the approach of fixing the number of colluders and the number of recovered movies and trying to find codes to support an optimal number of devices/users for a given number of variations of each movie. For example, the code shown in [17] either has too few codewords (accommodates a small number of devices) or the number of variations is too large (requires too much space on the disc). In the AACS context, a traitor tracing scheme must first meet the two requirements on the number of variations and the number of devices, then its goal is to minimize the number of recovered movies to detect unknown number of colluders.

In existing literatures, the scheme shown in [20] can meet the first two requirements. In this scheme, basically for each movie, there is an ”inner code” used to assign the different variations at the chosen points of the movie; it effectively creates different movie versions. Over a sequence of movies, there is an ”outer code” used to assign movie versions to different players. Both assignments can be random or systematic. For example, one can use a Reed-Solomon code for both the inner and outer code. Suppose there are 16 variations created at each of the 15 points in the movie. Their scheme will create 256 versions in which any two versions will be guaranteed to differ at least 14 points. Once the ”inner code” creates the multiple movie versions (e.g., 256), each player is assigned one of the 256 versions for each movie in a sequence of 255 movies. A Reed-Solomon code can create 256^4 codewords (thus billions players) with any two players differ at least 252 movies. By concatenating the two levels of codes, the assignments managed to avoid having a big number of variations at any chosen point but can still accommodate the billions of devices. These parameters can be good choices for AACS.

As mentioned above, the ”outer code” is used to assign the movie versions to each player. For real use, the ”outer code” can be used to assign the movie version keys to each player. Those keys will be burned into the player at manufacturer time and will not get updated afterwards. These keys are called “sequence keys” in the AACS specification. For example, each device is assigned a set of 255 keys, corresponding to the 255 movies in the sequence. Each key comes with 256 versions corresponding to the 256 movie versions created from the ”inner code”. During playback time, the device can use the sequence key for a movie to obtain the actual variation encrypting keys for each segment. More details are referred to [20].

The first two requirements have to do with the assignment step in a traitor tracing scheme. While AACS adopts [20] for its assignment step, the third requirement on the traceability cannot be met with the scheme [20], measured by the number of recovered movies needed in order to detect traitors involved in a coalition. In fact, the solution to the traceability problem has more to do with the actual traitor/coalition detection step.

As we mentioned earlier, there is one thing common with all the existing schemes including [16][17] and [20] on the tracing step. For each device, they calculate the number of matching that the observed pirate copies have in common with the versions assigned to that device. When the traitors in a coalition collude together in the pirate attack, these schemes are defined to detect and incriminate the one traitor who has the most matching. In fact if we use the highest score tracing approach on the above assignment, as shown in [20], for a coalition of 10 traitors one of them can be detected after 255 movies. This is not enough for practical use. In fact, the authors for [20] called for a more efficient probabilistic tracing approach.

3 Our traitor/coalition detection algorithm

Our algorithm works with both random and systematic assignment of the keys to devices. In this paper, for the sake of simplicity, we will just assume that the licensing agency assign the keys uniformly at random instead of using a Reed-Solomon code.

First of all, we want to make clearer of what we mean by ”coalition”. To our coalition detection algorithm, a coalition exists if illicit movies are coming from many players and we cannot otherwise determine which movies are coming from which players. For example, we recover re-broadcasted movies in a file sharing network. It does not mean that the people in the coalition are organized. It does not even mean that they know about each other’s existence.

As mentioned earlier, traitor tracing schemes in literatures have been mostly focused on the assignment step. The actual detection algorithm is simple and straightforward: you take your sequence of recovered movies, and simply score all the devices based on how many movies match with what each device has been assigned. You incriminate the highest scoring device. Traitors are therefore detected one by one. But why not detect every member in the coalition all together? The classic one-by-one method has some obvious advantages:

1. It seems easier.
2. The number of coalitions of a certain size is exponential in the number of users in the system. For example, if there are 1,000,000,000 devices/users in the world, there are roughly 500,000,000,000,000,000 pairs of devices (i.e., coalitions of size 2).
3. It seems essential against the "scapegoat" strategy. In this strategy, the coalition sacrifices a few devices and uses them heavily while using the others lightly, to keep some in reserve. Note that even without the scapegoat strategy, simulation results usually show some unlucky innocent devices intermixed with guilty players when the devices are scored in the classic way.

It may seem counter-intuitive, but we believe it is easier to find the entire coalition than to sequentially find one individual traitor, disable him and find another one. It turns out that it is much less likely that coalitions appear by random chance, than that individual player randomly has high score. An example can informally illustrate the underlying idea. Suppose there are 4 people involved in a colluding attack, and we have a random sequence of 20 recovered movies. Each movie originally has 256 variations of which a given player only plays 1. The attackers wish to see that high scoring device can happen by chance. If the four attackers are using round robin, each guilty player will evenly score 5. Can we incriminate any player that share 5 movies with the recovered sequence? No, there will be about 15 completely innocent players scoring 5 or greater due to chance alone. What can you do then? You have to recover more movies before you can incriminate any player. In general, with N players and q variations for each movie, the expected number of individuals who can score x among m movies are:

$$N * (1/q)^x * \binom{m}{x} \tag{1}$$

However, the above 4 guilty players together can explain all the movies in the sequence. What is the chance that a coalition of size 4 might have all the variations in the sequence? The answer is roughly 0.04. In other words, while there are plenty of players that can explain 5 movies, it is unlikely that any four of them can "cover" all twenty movies. If we find four players that do cover the sequence, it is unlikely that this could have happened by chance. It is more likely that that some devices in the coalition are indeed guilty.

On the other hand, the attackers may use scapegoat strategy. Some player is used heavily, for example, score 9 or 10. The traditional approach can correctly identify him, but it is hard to find the lightly used player and the true coalition size. Our new tracing algorithm can nonetheless find the other members in the coalitions and find out the coalition size.

In section 3.1, we will show how we find a coalition to explain the recovered movies. After we find the suspect coalition, in section 3.2 we will show how we identify the actual guilty players in the suspect coalition and filter out the innocent ones.

3.1 Finding a coalition

Let us formalize the above intuition a bit more. If there are N players, and a sequence of m movies are selected, each movie having one random variation out of q, the expected number of coalitions of size T are:

$$\binom{N}{T} * (1-(1-1/q)^T)^m \tag{2}$$

If the expected number of coalitions is less than 1, this formula also gives an upper bound on the probability that a random sequence of m movie variations is covered by a coalition of size T.

In AACS context, as a sample parameter, $q = 1024$ and a reasonable $T = 40$. If T is in fact noticeably less than q, a simplification of this is a close upper bound:

$$\binom{N}{T} * (T/q)^m \tag{3}$$

The problem of finding a coalition of players that covers a sequence of movies is equivalent to a well-known problem in computer science called Set Cover. It is NP hard. Any set cover algorithm can be used here. But we find there is even no need to use a much elaborated set cover algorithm. Not only that computational time is not much an issue for AACS, but also in reality the calculation time is very reasonable for the parameters that AACS is concerned with. For example, using the simple set cover shown below, to detect coalitions that cover 20 movies, it takes about 5 seconds on a Thinkpad T30.

Assume the licensing agency has observed a sequence of movies and determined the particular variation (the "symbol") in use for each. We also introduce the parameter k, the number of symbols that would probabilistically identify a single player. For example, k could be set to $log_q N$, where N is the total number of players.

The following recursive procedure *COVER*, if given a suspected number of traitors T and a list of the m encoded symbols discovered, returns true if and only if there is at least one coalition of size T that can explain the observed symbols:

1. If $T * k$ is greater than the number of symbols, print "many" and return true.
2. Calculate the minimum number of symbols that the largest-scoring traitor must have:
$$min = \lceil \frac{m}{T} \rceil$$

3. For each possible combination of k symbols, calculate whether the single player assigned to that combination covers '*min*' number of symbols. If it does, perform the following:
   a. If $T = 1$, print the player ID and return true.
   b. If $T > 1$, recursively call *COVER* passing the symbol list after removing all the symbols from the suspect player and with $T = T - 1$.
      i. If the recursive call returns false, continue to loop through the other combinations.
      ii. If the recursive call returns true, print the player ID and return true.
   c. If all combinations have been checked, return false.

The tracing algorithm assumes that the size of the coalition is unknown, and proceeds to calculate both the size of the coalition as well as the actual players involved. Below is the method that uses the above procedure *COVER* (or any other Set Cover procedure):

1. Set T = 1.
2. Run *COVER*.
3. If *COVER* returns true, exit.
4. Otherwise set $T = T + 1$ and loop to step 2.

Eventually the procedure must exit at step 3. Why? Once the number of movies is less than $T * k$, *COVER* is guaranteed to return true (see step 1 in *COVER*). But the interesting thing happens if you exit "early". In this case, you have found a coalition, and you can calculate the probability that a larger completely different coalition could have incriminated this coalition of size T, as explained in Lemma 1.

3.2 Identify guilty individuals in the found suspect coalition

Once we have found a coalition, who in the coalition should we incriminate? What is the chance that some of the players in the purported coalition of size T might be actually innocent, being victimized by a scapegoat strategy that is hiding a few lightly used guilty players? We calculate this as follows:

For each combination of T players, perform the following steps:

1. Temporarily assume that the players in the particular combination are guilty.
2. If the number of players in this combination is c, subtract c from T
3. Temporarily subtract from the list of movies all the movies that can be explained by this combination of players.
4. Use the formula 2 above using the new number of movies m and T, to evaluate the probability that the remaining players are completely innocent. If the formula yields a number greater than 1, assume the probability is 1.

When this procedure has ended, there will be a list of all possible combinations of players together with the chance that the remaining players are innocent. If some of these combinations indicate that there is a good chance that a player is innocent under those circumstances, the licensing agency would be well advised not to take action against the player (yet). On the other hand, some players will seem guilty under all combinations. In other words, the license agency can use the minimum guilty probability of the each player under all combinations as the probability of guilt of the player. In general, players that score higher in terms of the number of movies they could have encoded are also more likely to show up as guilty after the procedure. It is also reassuring that after this procedure any player that is identified only as "many" in the *COVER* procedure will show up as likely innocent.

Note it is possible that two of the players in the coalition may have a high overlap in movies. In this case, the procedure above might reveal that if player A is guilty, there is a good chance that player B is innocent, and vice versa. In this case, the licensing agency would be well advised to avoid making a decision about either of them until more movies have pointed to one or the other. Note that using the "*min*" probability rule, both players show up as likely innocent for the time being. However, the policy used by the licensing agency is outside of the scope of this paper. This algorithm provides the necessary tool to the licensing agency: a short list of potentially guilty players and probability of their actual innocence or guilt.

We now discuss a few optimizations. Before calling *COVER* the first time, it is usually faster to pre-calculate the $\binom{m}{k}$ potential players. Then, in step 3 of cover, you simply iterate through the pre-calculated list, seeing if each player is still a candidate under the current circumstances. Determining which player corresponds to particular list of k symbols can often be optimized. It is always possible to exhaustively search through all the players to see which one is indicated, but this can be obviously sped up by well-known techniques like table look-up and hashing. Furthermore, if the encoding method used is a linear code, as it was shown in our previous paper [20], it is possible to identify the player by algebraic means. For example, each list of k symbols defines k equations in k unknowns, which can be solved by Gaussian elimination.

4 False positive

Our tracing algorithm assumes that the size of the coalition is unknown, and proceeds to calculate both the size of the coalition as well as the actual players involved. If the size of the coalition is known from other sources, the answers may be exact; otherwise, the answer is always probabilistic. The problem is, from the attackers side, they do not know what sequence would incriminate an innocent player, so they are just guessing. We can make the probability they guess correctly arbitrarily small by just collecting more movies. The following lemma shows the false positive rate in our detection.

Lemma 1. *Assume that a coalition of guilty players cannot deduce the movie assignment of any other player in the world, for a coalition C, $|C| = T$, found by algorithm* COVER*, the probability that every member in coalition C is innocent is* *bounded by formula 2. In other words, the formula gives the false positive probability in the detection.*

Proof: Imagine that the process of assignment is the opposite of the way it works in real life: instead of starting with the assignment of variations to the population, the coalition randomly picks their assignment and then picks the particular variations of m movies in any way they choose. Only then does the licensing agency, not knowing what the coalition has picked, assign the variations for the remaining innocent players randomly. The chance that this assignment would result in a coalition of size T amongst the innocent players is clearly bounded by equation 2. And since there is no way to distinguish the "real life" case from the "thought experiment" case based on the player assignment (they are both equally random), the equation does represent the best that the attackers can do. □

The licensing agency can choose any acceptable value for the false positive rate. The smaller the false positive rate, the more pirate movies it needs to recover. We can get any kind of confidence level desired, but it will just take us more recovered movies to achieve. If the attack is ongoing, we always have the option of increasing our confidence by recovering more movies. In general, for each movie recovered, our confidence that the guilty players are, in fact, guilty is increased by roughly q/T. Since our entire tracing is probabilistic, we can factor in some false positives from the underlying watermarking technology (that is determining which variations were recovered) as well.

5 Tracing efficiency

From formula 3, we can calculate the number of movies m it takes for a coalition of size T to achieve any level of confidence (or false positive rate), for example, λ. We obtained a superlinear relationship between m and T.

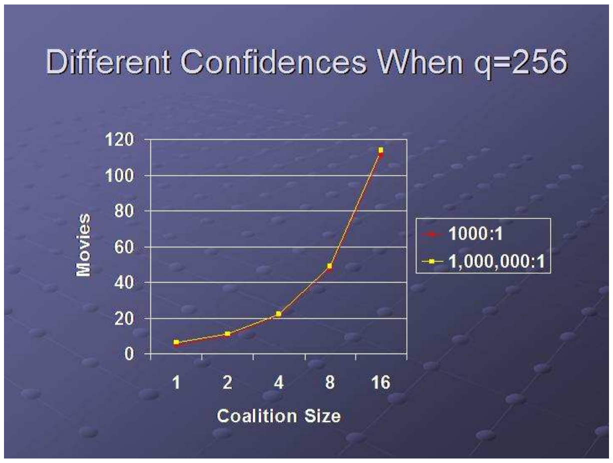
$$\binom{N}{T} * (T/q)^m = \lambda \tag{4}$$

Because N is much larger than T, $\binom{N}{T}$ can be approximated to be N^T. Solving the above equation gives us:

$$m = \frac{T * lnN - ln\lambda}{lnq - lnT} \tag{5}$$

For the parameters of our choice for AACS, it is easy enough to use a spreadsheet on the formula 3 to show the relationship among these numbers. The two graphs below show this relationship when the number of device is 1 billion.

Interestingly, it takes almost the same number of movies (roughly $6T$) to achieve a super high confidence (below 0.0001%) as it does to achieve a moderately high confidence (below 0.1%)



Now let us do some comparison with existing approaches. Of course in AACS context it is difficult to deploy a dynamic traitor tracing scheme like [15] because AACS has to assign the sequence keys to burn into devices during manufacture time and cannot easily update them afterwards. Among static schemes, a traceability codes is one of the traditional approaches that incriminates the highest score device, i.e. the device whose codeword is at the smallest Hamming distance from the pirated copy. Indeed a traceability code enables one to decode to the nearest neighbor of a pirate code and the nearest neighbor is deterministically a traitor.

Lemma 2. [5] Assume that a code $\mathscr{C}$ with length n and distance d is used to assign the symbols for each segment to each user and that there are t traitors. If code $\mathscr{C}$ satisfies

$$d > (1 - 1/t^2)n, \tag{6}$$

then $\mathscr{C}$ is an t-traceability-code.

In [20], it showed the tracing results based on the above formula when using the parameters of choice for AACS. It can deterministically identify traitors in a coalition of nine after recovering 256 movies. In contrast, for the same coalition size, our algorithm takes 56 movies and the false positive rate can be low at 0.0001%. Indeed [20] called for probabilistic tracing to improve efficiency as well as fit more with the reality that coalition size is unknown in advance.

As another one-by-one detection scheme, the static sequential traitor tracing scheme shown in [16] can detect T traitors with $T^2 + T$ movies. For the reasonable coalition size that AACS is concerned with, for example, a dozen to several dozens traitors, our superlinear results shown in Formula 5 is much more efficient.

Please also note that the probabilistic tracing we have is also different from the probabilistic tracing in [4, 5]. Their goal is to make the probability of exposing an innocent user as small as possible, while we try to make the probability of catching the actual traitor to be reasonably high.

6 Simulation results

We have also performed simple simulations to confirm the above analysis. Because of the nature of the probabilistic detection, it means some false positive. For a coalition of size 4, we know it takes about 22 movies to detect the traitors with very high confidence. Of course, to confirm a very low probability like that would take an unreasonably large number of simulations. Instead, we used a test with a larger false positive rate, namely a 20 movie sequence. We randomly picked a coalition of size 4, and create 20 pirate movies out of the chosen 4 traitors. We tried both random and round robin methods for the traitors' strategy. We confirmed (at the 95% confidence level) that equation 2 holds. Similarly, for a coalition size of 6, from the formula we know it takes about 34 movies to reach a confidence 0.005% false positive. We simulated using only 32 movies. After 100 simulations we tested, we found 6 cases that involve a completely innocent coalition, which is consistent with the bound from equation 2, which is 9.5%.

We also notice a slight difference of the behavior when we use round robin to create the pirate movies than when we use random selection. In the case of random selection, one player often contributes a lot. This partially explains why traditional score ranking could work to some extent against the random selection attacker strategy. But with our new tracing scheme, the other coalition members are nonetheless found, unless they made a negligible contribution to the attack.

On the other hand, in the case of round robin, the movies are contributed evenly from the attackers. It is hard to incriminate the highest scoring player in this case. For example, in the case of a coalition of size 4 and with 20 movies, all 4 players explain 5 movies. In our simulation, in most cases, the new tracing algorithm found the exact one coalition that together can explain all 20 movies. Once again, this explains why our new tracing algorithm is more efficient than the traditional approach.

7 Conclusions

In this paper, we study the problem of traitor tracing for re-broadcasting attack where 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 designed an efficient traitor detection algorithm for AACS (Advanced Access Content System) copy protection standard for the next generation of high-definition DVDs. The efficiency is measured by the number of recovered movies it takes to identify all the traitors in the coalition.

We take into considerations of some realities that have been overlooked in existing work. We designed a probabilistic tracing scheme that is closer to the real world situation more than deterministic tracing. It also achieves super linear traceability, much more efficient than existing approaches. Different from existing approaches which try to detect traitors one by one, we detect multiple traitors in the coalition together. This idea enables faster tracing with less recovered content, at the cost of higher computational overhead. We take advantage of the fact in reality this tradeoff is gladly made.

The superior traceability achieved by the algorithm described in this paper made its commercial adoption by AACS to protect the next generation DVDs. In the future, we will continue to improve its traceability, not only theoretically, but also by taking into consideration of real implementations. Technically we are interested in improving the filtering algorithm. We would also like to consider the case when the coalition size is large to anticipate new types of attacks enabled by future new technologies.

References

1. http://www.aacsla.com
2. 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.
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. M. Naor and B. Pinkas, ”Efficient Trace and Revoke Schemes”, Financial Cryptography’2000, Lecture Notes in Computer Science, Vol. 1962, pp. 1-20.
7. D. boneh, C. Gentry and B. Waters, ”Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys”, Crypto’05. pp.258-275.
8. D. Boneh, A. Sahai and B.Waters, ”Fully Collusion Resistant Traitor Tracing With Short Ciphertexts and Private Keys”, EuroCrypt’06, pp.573-592.
9. D. Boneh and M. Franklin, ”An efficient public key traitor tracing scheme”, Crypto’99. LNCS 1666, pp.338-353.
10. D. Boneh and B. Waters, ”A collusion resistant broadcast, trace and revoke system”, ACM Communication and Computer Security, 2006.
11. K. Kurosawa and Y. Desmedt, ”Optimum traitor tracing and asymmetric schemes”, EuroCrypt’98, pp.145-157.
12. H. Chabanne, DH. Phan and D. Pointcheval, ”Public traceability in traitor tracing schemes”, Eurocrypt, 2005, pp.542-558.
13. D.R.Stinson and R. Wei, ”Key preassigned traceability schemes for broadcast encryption”, ACM SAC’98, 1998.
14. E. Gafni, J. Staddon and Y.L.Yin, ”Efficient methods for integrating traceability and broadcast encryption”, CRYPTO’99, Lecture Notes in computer Science, Vol. 1666, 1999, pp. 537-554
15. A. Fiat and T. Tassa, “Dynamic traitor tracing,” *Crypto’99, Lecture Notes in computer science*, Vol. 1666, pp354-371. Springer-Verlag, Berlin, Heidelberg, New York, 1999.
16. R. Safani-Naini and Y. Wang, “Sequential Traitor tracing,” *IEEE Transactions on Information Theory*, 49, 2003.

17. Tran van Trung and Sosina Martirosyan, ”On a class of Traceability Codes”, *Design, code and cryptography*, 31(2004), pp 125-132.
18. J. N. Staddon, D.R. Stinson and R. Wei, “Combinatorial properties of frameproof and traceability codes,” *IEEE Transactions on Information Theory*, 47 (2001), 1042-1049.
19. D.R.Stinson and R. Wei, “Combinatorial properties and constructions of traceability schemes and frameproof codes,” *SIAM Journal on Discrete Mathematics*, 11:41-53, 1998.
20. H. Jin, J.Lotspiech and S.Nusser, ”Traitor tracing for prerecorded and recordable media”, ACM DRM workshop, Oct. 2004.
21. G. Tardos, ”Optimal Probabilistic fingerprint codes”, in proceedings of the *Theory of Computing*, pp. 116-125, June 9-11, 2003, San Diego, CA.