RJ10394 (A0611-015) November 1, 2006
Computer Science

IBM Research Report

Renewable Traitor Tracing:
A Broadcast, Tracing and Revoke System for Anonymous Attack

Hongxia Jin, Jeffery Lotspiech
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Renweable traitor tracing: a broadcast, tracing and revoke system for anonymous attack

Hongxia Jin, Jeffery Lotspiech

IBM Almaden Research Center
San Jose, CA, 95120
{jin,lotspiech}@us.ibm.com

Abstract. In this paper we design encryption schemes for combating piracy in mass distribution of copyrighted materials. When a pirated copy of some copyrighted material or an illegal decryption device is found, the scheme could identify at least one of the real users (traitors) who participate in the construction of the pirated content/device. More importantly, once the attackers are identified, the keys that have been used in the piracy can be revoked.
We start by briefly showing our traitor tracing scheme that can be used to defend against anonymous attack where the forensic evience is the pirated content or per-content decryption key. Content is divided into multiple segments and each segment comes with multiple variations. Each user can only play back one variation through the content.
We will then focus on integrating broadcast revocation capability into our tracing scheme. When traitors are identified, the enhanced scheme can revoke and exclude the decryption keys used by the traitors during piracy. The revocation information will be included in the newly released content, which will disallow the devices owned by traitors to playback the newly released content.
We believe this renewability is essential for a tracing scheme to be useful in a real system. Revocation is a natural next step after tracing to make it a complete system. Our trace and revoke scheme has been the first commercial use of a traitor tracing technology within the AACS [1] (Advanced Access Content System) content protection standards for next generation high-definition video optical disc.

Keywords: *Content protection, traitor tracing, broadcast encryption, anti-piracy*

1 Introduction

This paper is concerned with the protection of copyrighted materials. There are many business scenarios that content needs to be distributed through broadcast channels. Examples of these business models include pay-TV systems (Cable

[1] AACS is in business available for licensing now, see press release at http://www.aacsla.com/press

companies) or movie rental companies like Netflix, and massively distributing prerecorded and recordable media. A broadcast encryption scheme can be used to protect the content copyright and make sure the content can only be recovered by a previliged group of users. In this scheme, each user/decoder is assigned a set of keys that can be used to decrypt the broadcasted content in encrypted form. This set of keys are usually called device keys. In real use, oftentimes hybrid encryption is used, where an actual content encryption key (called media key) is chosen and the media key is encrypted by different device keys again and again and put in a structure called Media Key Block (MKB). Only those enabled devices can decrypt the MKB and obtain the correct media key to decrypt the content. The disabled devices cannot decrypt the MKB correctly. A group of attackers can come together and build a clone decoder with built in decryption keys that can decrypt the content. They can sell the clone decoder for free or a profit. Or they can publish a program on the Internet that can decrypt the content. We call this DeCSS type of attack a "clone decoder attack".

In this paper we are concerned with a different type of attack, which is anonymous. For example, an attacker re-digitizes the analogue output from a compliant device and redistribute the content in the clear. In this case, the only forensic evidence is the unprotected copy of the content (content attack). Or, the attackers may compromise the tamper-resistance of one or more players to extract and redistribute the actual content decryption keys (media key). In both attack scenarios, the attack is anonymous. To defend against these types of anonymous attacks, one needs different versions of the content and decryption keys (media key) for different users. Of course, then they can then set up a server that sells decryption keys on demand (key attack).

For the rest of this paper, we will assume that the content protected is copyrighted digital movies. We do this for concreteness, and because we were inspired by the AACS application. This movie application is true of many different one-to-many content distribution channels, in that it is generally infeasible to prepare and send individualized (e.g., watermarked) movies to each user. On the other hand, it is also infeasible, usually for security reasons, to customize each copy at the receiving end[2]. A feasible technical approach is to choose certain points in the movie and create different variations for each of those points. Each variation is differently encrypted. The movie is thus augmented by all the variations. Each user receives the same bulk-encrypted movie. However, each user can only decrypt one of the variations at each point. In other words, each recipient would follow a different path through the variations during playback time. It effectively creates multiple versions of a movie. Over time, when recovering enough pirate movies, it may be possible to detect the devices in a copyright attack by examining the variations recovered in the unauthorized copies of the movies. The devices/users used in the pirate attack are called "traitors" or "colluders".

The technology for identifying traitors is termed "traitor tracing" in the literature. The term has been used for both "clone decoder attack" and "anonymous

[2] Unauthorized copies generally imply a break the correct operation of the client. How can a broken client be expected to correctly generate the customizing marks?

attack". It refers to a way of identifying the source of users who leaked their keys (for redistribution or builing clone device) or constructed unauthorized copies of digital content. [4] [5] [16] are all on clone decoder attack. [9, 8, 10] are on anonymous attack. Of course in this paper we are mainly concerned with anonymous attack.

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

1. Assign variations or keys for the content to devices.
2. Based on the recovered version keys/content, trace back to the traitors.

A tracing scheme is static if it pre-determines the first step before the content is broadcast and does not change afterwards. A traitor can be identified when enough copies of keys/content are recovered. On contrast, if the first step can be updated based on observed pirate keys/content, the scheme is dynamic. Fiat and Tassa introduced a dynamic traitor tracing scheme [8] for anonymous attack. It involves realtime computation to decide the new assignment for the first step.

A tracing scheme is deterministic if it detects at least one exact traitor without any chance of incriminating an innocent user, otherwise it is probabilistic. The first traitor tracing scheme was proposed by Chor, Fiat, Naor and Pinkas in [4] for clone decoder attack. The traitor tracing schemes in [4],[5] are static and probabilistic. There is also a sequential traitor tracing[9] [10] which is static and deterministic. They are for anonymous attack. More formal analysis of its traceability is shown in [11], [12]. For the static assignment, it can be random or systematic. The traitor tracing schemes in [4],[5] randomly assign the decryption keys to users before the content is broadcast. The main goal of their scheme is to make the probability of exposing an innocent user negligible under as many real traitors in the coalition as possible. The traitor tracing schemes in [9] [10] used systematic assignment.

In current state of art, revocation seems to be natually considered inside a broadcast encryption scheme [3]. A broadcast encryption scheme is used to transmit a message from a center to a previliged set of users and able to exclude certain users from recovering the message. One of the criteria used to evaluate a broadcast encryption scheme is its renewability, namely, the revocation capability, measured by the number of users that can be excluded from recovering the message and the cost associated with the revocation, including the message length and the number of encryptions needed. In a typical broadcast encryption scheme, one is given as input the set of devices that needed to be excluded from recovering the content. But of course how does the agency know who should be revoked? It is most likely the outcome of a tracing traitor scheme. Even though the traceability and renewability seem to be naturally connected, these two attributes are orthogonal and have been mostly studied separately. For example, most recently, [15] showed a public key broadcast encryption scheme but without traceability, and the same authors in [16] showed a traitor tracing scheme based on the similar idea. Integration of tracing and revocation yield trace-and-revoke system. The above authors also come up with a combined public key trace and revoke system in [20]. While [6] only consider combinatorial constructions, the schemes in [7] is more general. Both [7] and [20] were considered

in the context of public key tracing. The revocation and tracing scheme shown in [2] also attempts to seamlessly integrate these two attributes together, but in symmetric key setting. One thing they are all in common is that they are all concerned with clone decoder attack.

For clone decoder attack, most popular way is to use black-box tracing. In black-box tracing, carefully crafted testing messages are fed into the box to see if the box decrypts. There is a multi-time tracing in [17] but was later shown in [18, 19] to be insecure. The flaw in the design of that scheme is that it doesnot prevent the traitors from generating an untraceable combination of their keys, which can serve as a decryption key. Unfortunately supporting multi-time tracing is a must in real use.

1.1 Main results

The authors have been involved in providing a solution to defend against the anonymous attack for the (Advanced Access Content System) content protection standards for next generation of high-definition optical DVDs. A practical scheme cannot require much extra bandwidth/storage overhead (for example, 10% of normal usuage). It has to support large number of users (for example, a billion). For a tracing scheme to be useful, we believe revocation must be able to happen after the piracy is identified and render the pirate decryption keys useless. It is also critical for the scheme to support continued multi-time tracing over the lifetime of the system. Basically for a scheme to be useful in practice, it demands a complete broadcast encryption, trace an revoke system for anonymous attack.

Note that there is big difference when coming to design a trace-and-revoke system for clone decoder attack and anonymous attack. For clone decoder attack, the message in the newly broadcasted content only needs to disable the compromised keys. For tracing purpose, separate testing messages are generated and fed into the clone box in order to trace keys contained in the clone box. However, for anonymous attack, it demands the same message containing the revocation information in the newly broadcasted content to not only revoke the compromised keys but also enable continued tracing of new traitors. Again, it demands a combined broadcast, trace and revoke system.

The main contribution of this paper is that we have designed the first broadcast, trace and revoke scheme with integrated broadcast, revocation and renewal capability for anonymous attack. We start by a traitor tracing scheme that can identify traitors in an anonymous attack. On top of this tracing scheme, we then go on adding revocation capability similar to the design of Media Key Block in a broadcast encryption system. The traitor tracing keys serve the role of the device keys in a broadcast encryption system, and the actualy content encrypting keys in our traitor tracing scheme serve the role of the media key in a broadcast encryption system. Of course, there are multiple versions of content encrypting key in our scheme while there is only one media key in a normal broadcast encryption system. Unfortunately, the above combined scheme cannot support multi-time continued tracing. It is possible that the license agency does

not gain any useful information to continue detecting the unexposed traitors. We idenfied the potential security problem and improved our scheme to support multi-time tracing. Moreover our scheme uses symmetric keys not public keys to avoid computational overhead. Adding renewability and continued traceability was a crucial enabling factor for this first time large scale commercialization of a tracing traitor technology.

In rest of the paper we will start by describing the traitor tracing scheme in Section 3, and adding renewability into the scheme in Section 4. We will then talk about the security issues to support multi-time tracing and show our improved scheme in 5. We analyze the revocation capability and continued traceability in Section 6. We conclude in Section 7 with future work.

2 Pirate model

There are two well-known models for how a pirated copy (be it the content or the key) can be generated:

1. 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.
2. Given two variants v_1 and v_2 of a movie segment ($v_1 \neq v_2$), the pirate can generate any valid variant v_i out of v_1 and v_2

The second model, of course, assumes the attackers have more power. As shown in [14], when using this model, the lower bound of the number of movies it takes to detect traitors in a coalition of T is T^2. We believe the second model fits documents or software better than audio or video. Since in this paper we are concerned with audio and video, assuming the first model is acceptable in the AACS context. Indeed, this so-called marking assumption is often made by other traitor tracing schemes shown in the literature. Also, in a key attack, the first model says it is impossible to calculate a valid random cryptographic key from combining two other valid random keys–which is obviously true.

3 Tracing Scheme

In the AACS context, each movie is divided into multiple segments and each segment is augmented with multiple variations. As one can imagine, the variations takes extra space on the disc. 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. Of course the traceability of the scheme has to be reasonable.

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. In the AACS context, a traitor tracing scheme must first meet the above two requirements. Existing schemes do not fit here. For example, the code shown in [10] 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). Bringing the long-standing theoretical work to practice was the major effort we undertook in the AACS system.

We have designed a way to meet the above two restrictions as shown in [13]. We summarize its basic idea here. 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. For example, 16 variations are created at each of the 15 points in the movie, effectively generating 256 versions for each movie. For a sequence of movies, there is an "outer code" used to assign movie versions to different players. For example, each player is assigned one of the 256 versions for each movie in a sequence of 255 movies. By concatenating the two levels of codes, we managed to avoid having a big number of variations at any chosen point but can still accommodate the billions of devices we anticipate. The inner code and outer code assignments can be random or systematic. For example, both inner and outer code can use Reed-Solomon codes. This two level key assignment is the essence of the first step in our traitor tracing scheme for AACS.

As one can see, the device needs to know all the variation encryption keys for each movie in order to play back the movie. Using the above parameters, a device needs to know 255*15 keys. However, that does not mean a device needs to store that many keys. For the first step of the AACS tracing scheme, instead of directly assigning variation keys for each movie to the devices, a level of indirection allows us to assign movie version keys to the devices, from which the devices can obtain the actual variation decrypting keys for that movie. This level of indirection allows us to save space on the devices used to store the keys. In other words, only the "outer code" was used to assign keys to devices in the first step. 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 in the world. These keys are called "sequence keys" in the AACS specification. Many players will receive any given sequence key, but no two players will receive exactly the same set of 256 keys. These sequence keys are placed in the players at manufacturing time. For most types of players, it is impossible to dynamically update those keys. So our scheme has to be static.

Based on the pirate model we mentioned in Section 2, the attackers can construct the pirate copy of the content/key based on the available versions to them. When the license agency recovers pirated movies/keys, it tries to match the recovered movies/keys with the versions assigned to the devices and incriminates the one that has the most matchings.

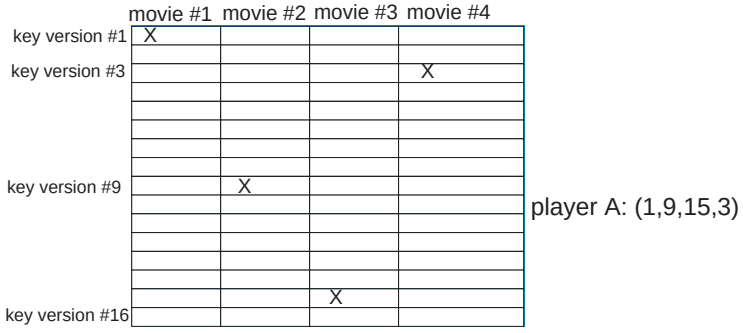

Fig. 1. Key assignment from a matrix

4 Revocation scheme

The AACS licensing agency will generate Sequence Keys organized in a large matrix. The matrix has 256 columns and not more than 65,536 rows. Each cell is a different Sequence Key. A single device has one key in each column. Thus, each device has 256 Sequence Keys. Attackers would prefer to use already-compromised Sequence Keys if they could, so that no new forensic information could be deduced by the licensing agency. Therefore, it is important that compromised keys are no longer usable by the attackers. The problem is that many thousands of devices might share a single compromised key. Therefore, revocation of a single key is impossible. On the other hand, revocation of a unique set of keys is very possible; in fact, that is precisely what the Sequence Key Block (SKB) achieves.

The fundamental principle is that no two devices have many keys in common, so even if the system has been heavily attacked and a significant fraction of the Sequence Keys is compromised, all innocent devices will have many columns in which they have uncompromised keys. The purpose of the Sequence Key Block is to give all innocent devices a column they can use to calculate the correct answer, while at the same time preventing compromised devices (who have compromised keys in all columns) from getting to the same correct answer.

In an SKB there are actually many correct answers, one for each variation in the content. For the purpose of explanation, however, it is helpful to imagine that a single SKB is producing a single answer. We will call that answer the output key.

As shown in Figure 2, the SKB begins with a first column, called the unconditional column. By column, we mean a column of Sequence Keys in the matrix

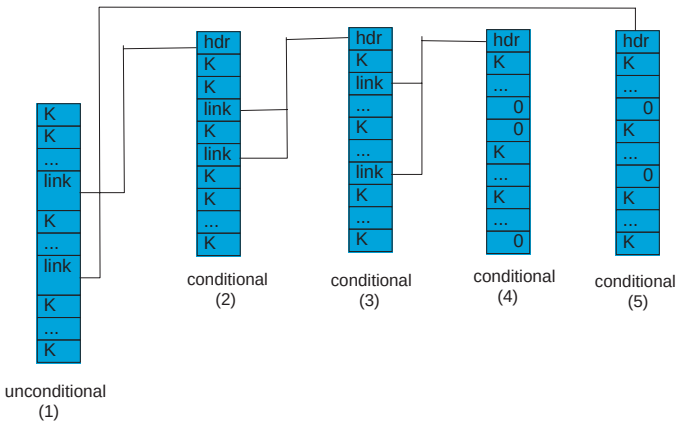

Fig. 2. Sample SKB

will be used to encrypt. (To be precise, the key used to encrypt is derived from the Sequence Key, not the Sequence Key itself.) The first column will have an encryption of the output key (denoted K in the figure) in every uncompromised Sequence Keys cell. Devices that do not have compromised keys in that column immediately decrypt the output key. Devices, both innocent and otherwise, that do have compromised keys instead decrypt a key called a link key that allows them to process a further column in the SKB. To process the further column they need both the link key and their Sequence Key in that column. Thus the subsequent columns are called conditional columns because they can only be processed by the device if it were given the necessary link key in a previous column.

The subsequent additional conditional columns are produced the same way as the first column: They will have an encryption of the output key in every uncompromised Sequence Keys cell. Devices with a compromised key will get a further link key to another column instead of the output key. However, after some number of columns depending on the actual number of compromised keys, the AACS licensing agency will know that only compromised devices would be getting the link keyall innocent devices would have found the output key in this column or in a previous column. At this point, rather than encrypting a link key, the agency encrypts a 0, and the SKB is complete. All innocent devices will have decrypted the output key, and all compromised devices have ended up decrypting 0.

How do the devices know they have a link key versus the output key? The short answer is they do not, at least not at first. Each conditional column has a header of known data (DEADBEEF16) encrypted in the link key for that column. The device decrypts the header with the key it currently has. If the header decrypts correctly, the device knows it has a link key and processes the column. If it does not decrypt correctly, the device knows it has either the output key or a link key for a different column. When the device reaches the end of the SKB without decrypting 0, it knows it must have an output key. Note that this device logic allows the licensing agency to send different populations of devices to different columns by having more than one link key output from a single column. For example, in the figure, column (1) links to both column (2) and column (5). This flexibility can help against certain types of attacks.

The preceding description is the basics of an AACS SKB and described in a simplied version. In an actual AACS SKB there is not a single output key, but multiple output keys called Variant Data D_v.

The SKB is generated by the AACS license agency and allows all compliant devices, each using their set of secret Sequence Keys to calculate the Variant Data, D_v, which in turn allows them to indirectly decrypt a table that contains the actual movie variation encrypting keys for the playback path of the movie assigned to that device. If a set of Sequence Keys is compromised in a way that threatens the integrity of the system, an updated SKB can be released that causes a device with one or more compromised sets of Sequence Keys to calculate invalid Variant Data. In this way, the compromised Sequence Keys are revoked by the new SKB. In fact, if a device correctly processes an SKB using Sequence Keys that are revoked by that SKB, the resulting final D_v will have the special value 000000000000000000016. This special value will never be an SKBs correct final D_v value, and can therefore always be taken as an indication that the devices Sequence Keys are revoked. Device behavior in this situation is implementation defined. As an example, a device could exhibit a special diagnostic code, as information to a service technician. If at any point, a device calculates a D_v value of zero, it should discontinue processing of the SKB and may conclude that it has been revoked.

5 Multi-time tracing

The above solution provides a key management scheme that not only can trace traitors but also can revoke compromised keys. With the added revocation into the tracing scheme, it now has broadcast capability in addtion to traceability, in other words, it can allow only the non-revoked devices (a previledged subset of devices) to decrypt and playback the content. However, our goal is to fully integrate traceability and broadcast encryption. By that we mean the scheme needs to continue responding to pirate attacks after revocation and detecting new traitors as it goes. Unfortunately the above solution does not live up to that goal yet. When attackers donot collude, we are safe. Problem arises when they collude, in which case the above solution cannot maintain continued traceability.

As we know, each column in a SKB contains an encryption of the output key in every uncompromised sequence key's cell. More precisely, in every uncompromised sequence key's cell in each column, it contains an encryption of one of the variant data D_{vi} (0¡=i¡k), k is the number of different variant data in the system. Notice the same set of variant data are encrypted in each column, although they are distributed differently in different columns. For a particular variant data D_{vi}, it can be obtained from any column in the SKB. A uncompromised device will process SKB and obtains a correct variant data from the first column in SKB that it has a unrevoked key. However, when attackers collude and the system still has undetected attackers at large. The attacker can mix-and-match their revoked keys and unrevokd keys when processing SKB. In turn they have multiple ways to process SKB and get a valid variant data to play back the content. They can choose in which column they want to use a unrevoked key to get a valid variant data. It does not have to be in the first column. MOreover, different keys need to be used in different columns to obtain the same variant D_{vi}. When the license agency observes a pirate copy corresponding to a particular variant data D_{vi}, since it can be obtained from any column, the license agency has no way to know which key has been used in obtaining that variant data. The entire path that the undetected traitors goes through to process SKB can even look like from an innocent device or from a path that was never assigned to any device, thus untraceable.

To force the undetected traitors to reveal the keys they use when processing SKB, we must make sure each column gets different variations so that when recovering a key/variation, the scheme knows from which column it comes from. Only by observing that, the tracing scheme can continue tracing. Unfortunately that means the q variations have to be distributed among the columns contained in the SKB. Each column only effectively gets q/c variations where c is the number of columns in the SKB. It is clear that traceability degrades when the effective q decreases. When the number of columns c becomes big enough, the traceability degrades to so low that it basically becomes untraceable. The scheme is overwhelmed and broken in that case. As a result, that puts a limit on the revocation capability of the scheme.

5.1 Improved solution

In order to improve the above scheme and lift the limit on revocation capability, we have designed a two phase defense. In order to use this defense, the scheme assigns the sequence keys as shown in 3 instead of 1.

Bascially we used 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 that is uses to assign sequence keys to devices is now itself a two-level code. The first level 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.

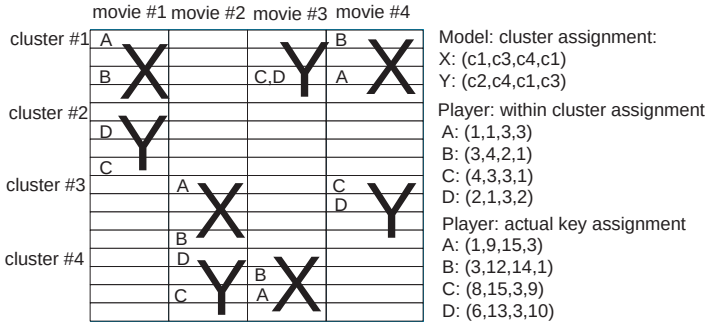

Fig. 3. Key assignment using "slots"

In AACS context, we anticipate about 4000 manufacturers/models. We divide all the rows in each column to have 64 clusters. Using Reed-Solomon code, $q = 64$, it takes $k = 2$ to accomodate 64^2 slots. Suppose we have $4K$ keys in each column, there will be 64 keys in each cluster. Again using Reed-Solomon code, $q = 64$, it takes $k = 3$ to accomodate 64^3 devices within each slot. The assignment can totally accommodate 64^5 devices. Each slot can be assigned to one manufacturer/model, A big manufacturer would, of course, overflow a single slot. He would just have more than one slot.

In the two-phase defense, If we are under attack, the first SKB's would determine the slot used in the attack (or slots, but that is unlikely). Since the slot is assigned from the first level by using Reed-Solomon code $q = 64$, $k = 2$, there are only 64 variations needed per column in this case. Recall the "inner code" generates totally 256 variations for each movie. One can use 4 columns in the SKB and there is no problem with dividing the 256 variations across 4 columns. Each column would get 64 variations, which is all we need per column. By Reed-Solomon code's property, it takes only two ($k = 2$) movies to uniquely detect the slot. The above attack scenario does not work here.

Once the licensing agency detects the slot, it can produce new SKB's that are only trying to detect the device within the slot. In the SKB, all other slots in the column(s) would go to a single variation that we would expect would never be recovered. We would use all the remaining variations within the single slot. Again, the above attack scenario is not much a problem, because we can get up

to 4 columns and still have unique keys for each variation. As long as we only need 4 columns in SKB, the above attack cannot work.

However, it still puts a limit on how many devices the scheme can revoke before it is overwhelmed. A single slot can be overwhelmed if it gets a lot of attacks. For example, if there are just 32 devices revoked from a single manufacturer, then 50% of the keys in the slot are compromised, and the SKB takes about 18 columns to winnow out that manufacturer's innocent devices. The 18 columns means the above attack can cause a problem, even with the two-phase approach. Because of this, a larger number of keys per column is preferred, for example, $16K$ keys/column would be a better number. These keys are still grouped by 64 clusters for slot purposes. Each slot can have 256 keys instead of 64 keys. Then we would be back down to 6 columns. Of course, anything more than 4 columns introduces the above attack problem, but it is still manageable.

6 Revocation capability and traceability

More formally,the revocation capability is calculated by the following formula. Suppose the number of rows in the matrix is m, p is the acceptable maximum probability for an innocent device to be revoked when revoking the actual guilty devices. r is the number of guilty devices to be revoked in SKB. c is the number of columns in SKB. The system still survives when the following holds.

$$(1 - (1 - 1/m)^r)^c < p \tag{1}$$

This formula can be used to determine how many columns c needed in a SKB when the licensing agency wants to revoke r devices. Due to the above attack scenario, there is a limit on the number of columns. We can also easily see that, the limit on the number of columns c in SKB induces a limit on the number of revoking devices that the system can survive. For example, suppose sequence keys are assigned from a matrix of $4096X256$, there are 4096 rows. We devide rows into 64 clusters for slot assignment. Suppose the acceptable rate for revoking innocent devices is $1/1,000,000$. Suppose there can be at most 18 columns in SKB. In the case of random hacking where we assume the attackers are distributed randomly from any slot and there is no evil manufacturers, the system can allow up to ??? guilty hacking devices, revoke them and can still get useful tracing information for continued tracing. In the case of evil manufacturers or most hacking occurs in one slot, breaking one slot breaks the entire system. In the above example, there are 64 cluster in a $4096X256$ matrix, there are 64 rows in each cluster for slot assignment. Substituting $m = 64$ gives us that the system can maximumly survive up to 32 revoked devices within a slot.

We have performed preliminary simulation on how many columns the scheme needs in its SKB in terms of the number of devices that needs to be revoked. It confirms the above formula 1.

We know the solution for the above attack has effectively decreased the q, thus the traceability. The traceability is measured by the number of movies it takes to recover in order to detect traitors in a coalition of T traitors. We show

in the graph 4 the different traceability on different q. It includes both the two-phase solution and the basic single-phase scheme. There is different traceability in the life of the system depending on how many devices are currently being revoked in the SKB. The more devices revoked, the more columns it needs in SKB, the smaller the effective q is, the worse traceability.

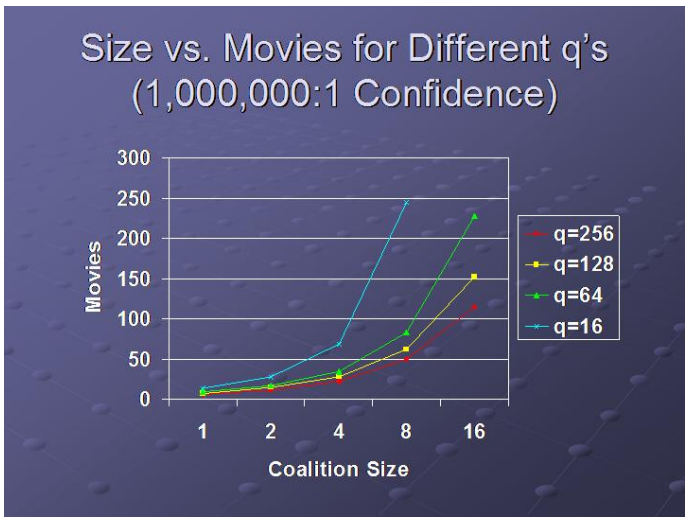

Fig. 4. Traceability with different q

7 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. In particular, we focus on distributing prerecorded movies in the context of Advanced Access Content System copy protection standard for the next generation of high-definition DVDs.

While researches have mostly separated consideration of traceability in a traitor tracing scheme and revocation capability in a broadcast encryption scheme, we believe these two capability should be integrated and considered in a single scheme for the scheme to be useful in practice. We have designed a complete broadcast encryption system that fully integrates tracebility and revocation capability together. We have designed the first trace-revoke-trace scheme that is of practical value in AACS context. We have analyzed its traceability and revocation capability. The revocation and continued tracing capability is one of the enabling factor for its first large scale commericialization of tracing traitor technology.

In the future, we will continue to improve its revocation capability as well as traceability, not only theoretically, but also by taking into consideration of real implementations. We are also interested in overcoming different barriers during its deployment.

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. 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
7. M. Naor and B. Pinkas, "Efficient Trace and Revoke Schemes", Financial Cryptography'2000, Lecture Notes in Computer Science, Vol. 1962, pp. 1-20.
8. 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.
9. R. Safani-Naini and Y. Wang, "Sequential Traitor tracing," *IEEE Transactions on Information Theory*, 49, 2003.
10. Tran van Trung and Sosina Martirosyan, "On a class of Traceability Codes", *Design, code and cryptography*, 31(2004), pp 125-132.
11. 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.
12. 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.
13. anonymous
14. G. Tardos, "Optimal Probabilistic fingerprint codes", in proceedings of the *Theory of Computing*, pp. 116-125, June 9-11, 2003, San Diego, CA.
15. D. boneh, C. Gentry and B. Waters, "Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys", Crypto'05. pp.258-275.
16. D. Boneh, A. Sahai and B.Waters, "Fully Collusion Resistant Traitor Tracing With Short Ciphertexts and Private Keys", EuroCrypt'06, pp.573-592.
17. K. Kurosawa and Y. Desmedt, "Optimum traitor tracing and asymmetric schemes", EuroCrypt'98, pp.145-157.
18. D.R.Stinson and R. Wei, "Key preassigned traceability schemes for broadcast encryption", ACM SAC'98, 1998.

19. D. Boneh and M. Franklin, "An efficient public key traitor tracing scheme", Crypto'99. LNCS 1666, pp.338-353.
20. D. Boneh and B. Waters, "A collusion resistant broadcast, trace and revoke system", ACM Communication and Computer Security, 2006.