TRAITOR TRACING FOR ANONYMOUS ATTACK IN CONTENT PROTECTION

Hongxia Jin
IBM Almaden Research Center, San Jose, CA, U.S.A.

Keywords: Traitor Tracing, Content Protection, Anti-piracy, Anonymous Attack, Broadcast Encryption.

Abstract: In this paper we take a closer look at traitor tracing in the context of content protection, especially for anonymous attack where the attackers pirate the content and re-distribute the decrypted plain content. When the pirated copies are recovered, traitor tracing is a forensic technology that can identify the original users (called traitors) who have participated in the pirate attack and involved in the construction of the pirated copy of the content. In current state-of-art, traitor tracing scheme assumes a maximum coalition size of traitors in the system and is defined to detect one traitor, assuming the detected traitor can be disconnected and tracing just repeats with the remaining traitors. In this position paper we argue this definition does not sufficiently reflect the reality where a traitor tracing technology is used to defend against piracy especially in the context of content protection. We believe a traitor tracing scheme should deduce the active coalition size and should be defined to detect all active traitors even taking into consideration that found traitors need to be technically disabled. We believe the traditional definition misleads in the design of an efficient and practical traitor tracing schemes while our definition much better fits the reality and can lead to design of efficient traitor tracing schemes for real world use.

1 INTRODUCTION

Content protection for copyrighted materials is all about making sure the materials are only accessible to user who are authorized to access. Pirated attackers want to bypass the restrictions and enable access of the content illegally. Business scenarios include pay-TV systems, NetFlix or massively distributing physical media like DVDs. The success of these types of business scenarios hinges on the ability to make sure the content is only accessible to authorized (paying) customers. Indeed piracy is one of the biggest concerns in entertainment industry. Digital copies are perfect copies. In a broadcast encryption (Fiat and Naor, 1993) based content protection system, each device (also called decoders, users) is assigned a unique set of decryption keys (called device keys). It defines a key management scheme that assigns keys to devices and encrypts the content that can guarantee that only compliant devices can decrypt the content, without requiring authentication of the device. Furthermore, because the distributed content is oftentimes large, for example, a movie is about 2G bytes, in order to save space or bandwidth during content distribution, hybrid encryption is usually used. For example, a content encrypting key (sometimes called media key or title key) is randomly chosen to encrypt the content; and the media key itself is then encrypted by compliant device keys again and again while the non-compliant device keys are used to encrypt only garbage string instead of the valid media key. The bulk encrypted media key is put as a header into the distribution package and distribute together with the encrypted content. During playback, a compliant device can use one of its valid device keys to decrypt the header first to get a valid media key which in turn allows it to decrypt the content. The non-compliant devices can only decrypt to garbage and therefore cannot decrypt the content. There are different types of pirate attacks in the above content protection system. Every pirate attack enables one to access content illegally.

1. Pirates use compromised device keys to build a clone pirate decoder.
2. Pirates re-distribute content encrypting key (media key): pirates stay anonymous.
3. Pirates re-distribute decrypted content: pirates stay anonymous.

When the pirate evidence is found, forensic analysis can allow one to find out the source that the pirate copies come from. In literatures, the forensic technology to defend against piracy is termed as ”traitor tracing”. The source devices (users) that involved in the

Jin H. (2008).
TRAITOR TRACING FOR ANONYMOUS ATTACK IN CONTENT PROTECTION.
In *Proceedings of the International Conference on Security and Cryptography*, pages 331-336
DOI: 10.5220/0001929003310336

construction of the pirated copies are called traitors. For different types of pirated attacks, different types of traitor tracing schemes are needed.

In first pirate attack, the secret device keys are extracted from one or more compromised devices and put into a clone device that can decrypt the encrypted content. When a clone device is captured, in order to do traitor tracing (Chor et al., 1994; Naor et al., 2001), forensic testing materials are fed into the clone. In each forensic testing, the compliant device keys are partitioned into two subsets. Device keys in one subset are chosen to encrypt the valid media key while device keys in the other subset are chosen to encrypt the garbage to put into the header. Observing the forensic testing results which is play or not-play the content gives us information on which keys are inside the clone device. The traceability is defined to be the number of testings needed to detect traitors.

In an anonymous attack, the attackers decrypt the header and get the valid media key (or title keys) to decrypt the content. They can sell the decrypted content or serve the content decrypting keys on demand. In a traitor tracing scheme (Safani-Naini and Wang, 2003) (Jin et al., 2004) to defend against anonymous attack, content is encrypted differently for different devices. For example, content may be differently watermarked and encrypted. Of course preparing a different version for each different user is too costly, how to space-efficiently prepare and distribute the content is outside the scope of this paper. Readers refer to (Jin et al., 2004). What is relevant here in this paper is that different device will decrypt and play back content differently. Recovering the decrypted content or the content encrypting keys enables one to identify which keys the attackers have. The traceability of a traitor tracing scheme for anonymous attack is defined to be the number of recovered pirated copies of content or content encrypting keys needed to detect traitors.

In state-of-art, a traitor tracing is defined to be a scheme that can detect at least one traitor when a maximum number of traitors in the system is up to a certain number. The efficiency of a traitor tracing scheme is defined to be the number of recovered forensic evidences/results needed in order to detect one traitor. They also assume the discovered traitor can be disconnected in some way, and the tracing can simply be repeated for remaining traitors.

We believe this definition is inadequate and does not capture the reality of a traitor tracing scheme when actually used. In fact this definition does not help designing an efficient traitor tracing scheme. In our opinion, a traitor tracing scheme should be defined to probabilistically detect all active traitors involved in the pirate attack as well as deduce the size of active members in the coalition. The detection should also take into considerations that some discovered traitors will be technically disabled. We argue this in the context of anonymous attack, even though similar can be said to other attacks as well.

2 TRADITIONAL TRAITOR TRACING

As one can imagine, a traitor tracing scheme is all about how to assign the tracing keys (or content versions) to the devices and perform efficient forensic analysis after recovering forensic evidences such as keys and content versions.

For example, in traitor tracing for anonymous attack, content is differently watermarked and differently encrypted for different users. So, a traitor tracing scheme for anonymous attack usually consists of two basic steps:

1. *Assignment Step*: Assign different keys and content versions to devices,
2. *Traitor/coalition Detection Step*: Based on the recovered pirated content/keys, trace back to the traitors.

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 forensic evidences, be it the pirated copies of keys or content versions, and simply score all the devices based on how many the recovered copies 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 to detect one traitor at a time.
2. The identified traitor can always be disconnected, which potentially makes it faster to detect the second traitor given that the coalition size is smaller after disconnection of the previously detected traitor.
3. It seems it is much more complicated to detect guilty coalitions than individuals because the number of coalitions is much bigger than the number of individuals. In fact 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).

A further assumption made popularly is that once a traitor is detected, he can be disconnected and tracing can simply be repeated with remaining traitors. This is true sometimes when some legal means can be taken to actually disable the traitorous device.

Moreover, the research community define traitor tracing schemes to detect a traitor when the maximum size of the coalition is up to some threshold. This is because the actual coalition size is usually unknown.

Indeed the first traitor tracing scheme is defined in (Chor et al., 1994) to detect one traitor up to a maximum coalition size and is still the popular definition that is using now. A variant is deterministic or probabilistic tracing. A probabilistic tracing tries to make the probability of incriminating an innocent user as small as possible. Deterministic tracing can be seen in the popularly used approach called "traceability code".

A traceability code 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 2.1. (J. N. Staddon and Wei, 2001) Assume that a code 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 C satisfies

$$d > (1-1/t^2)n, \quad (1)$$

then C is an t-traceability-code.

We raise the following questions for arguments.

1. Is one-by-one detection really efficient?
2. When coalition size is unknown, is deterministic tracing even possible?
3. Can we deduce the active coalition size?
4. Is it reasonable to assume that the traitor can always be disabled in non-technical way?
5. Can we simply repeat tracing after disabling traitors?

3 OUR TRAITOR TRACING

We believe the above traditional definition of a traitor tracing scheme does not lead to the design of an efficient and practical scheme for real world use. In our opinion, a traitor tracing scheme should be defined to find all active traitors probabilistically and also deduce the active coalition size. In this section we will go through the above questions and argue why our position stands.

3.1 Is One-by-one Detection Efficient?

Notice that the ultimate goal of traitor tracing is to find all active traitors and disable them. While intuitively it seems easy and efficient to detect one traitor at a time, we have an anti-intuitive observation. it is easier to find the entire coalition than to sequentially find one individual traitor, disable him and find another one.

While the number of coalitions of a certain size is exponential in the number of users, it turns out that it is much less likely that coalitions appear by random chance, than that individual user 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 pirated copies of keys or content. Each key/content originally has 256 variations of which a given user(device) only knows 1. The attackers wish to see that high scoring device can happen by chance. If the four attackers are using round robin, each guilty user will evenly score 5. Can we incriminate any user that share 5 copies with the recovered sequence? No, there will be about 15 completely innocent users scoring 5 or greater due to chance alone. What can you do then? You have to recover more pirated copies of keys/content before you can incriminate any user.

However, the above 4 guilty users 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 users that can explain 5 movies, it is unlikely that any four of them can "cover" all twenty movies. If we find four users that do cover the sequence, it is unlikely that this could have happened by chance. It is more likely that some devices in the coalition are indeed guilty. Based on this counter-intuitive observation, we believe a more efficient forensic analysis for traitor detection algorithm would be to detect the entire coalition that can explain all recovered pirated copies and then filter out the innocent users from the found coalition if any. Readers see (Jin et al., 2008) for more details.

Now let us do a concrete comparison. Suppose each content/key comes with 256 variations and there are 1 billion devices in the system. The traditional

tracing based on Formula 1 can deterministically identify ONE traitor in a coalition of nine after recovering 256 pirated content/keys and it takes similar number of pirated copies to detect the second, and subsequent traitor. In contrast, for the same coalition size, the new algorithm based on detecting entire coalition can detect all active traitors using only 56 pirated content/keys and the false positive rate can be low at 0.0001%.

Of course, the attackers may use scapegoat strategy. Some device 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 device and the true coalition size. The new tracing can nonetheless find the other members in the coalition.

Again the ultimate goal is to detect and disable all traitors as fast as possible, we believe the traditional traitor tracing definition does not lead to the design of an efficient tracing scheme that can achieve the above ultimate goal efficiently.

3.2 Assume Coalition Size vs. Deduce Coalition Size? or Deterministic vs. Probabilistic?

We believe it is not practical to assume a maximum coalition size and perform deterministic tracing based on the assumed coalition size. Indeed, because the tracing agency rarely knows exactly how many devices are involved in the attack. As a result, the answers it gets are always qualified. For example, an answer might be as follows: "If N devices are involved, it must be exactly this N. However, different innocent coalitions of $N+M$ devices may have produced the same result." We will walk readers through a simple example to show how the actual tracing is done based on forensic evidence.

Suppose each content/key comes with 256 variations and there are 255 content/keys in the sequence. So each device is assigned 255 content/keys with one variation in each content/key. The assignment can be done using a systematic approach like error correcting code, for example, Reed-Solomon code. This approach can guarantee that any two users differ at at least 252 content/key assignment. This assignment can support 1 billion devices in the system. For any given content, $\frac{1}{256}$ of the devices (about 4 million devices) encode the content the same way. For a given three content, only $\left(\frac{1}{256}\right)^3$ of the devices (about 60 players) encode those content the same way. For a given four content, exactly 0 of the devices encode the content the same way. That is the essential property of the Reed-Solomon code assignment.

Let us take the case of an attack where only a single device X is being used. After recovering a single content/key, the license agency has four million devices that are potential candidates, including X. After the second or third recovered pirated content/keys, the number of candidates is reduced, but it is not until the fourth pirate content/key is recovered that the guilty device X positively identified– BUT only if it is known that only a single device is involved. Millions of pairs of devices could also have produced the four pirated content/keys.

By the time nine pirated content/keys have been recovered, the license agency knows there are no possible innocent pairs of devices. (By "innocent", we mean a pair that does not include the actual guilty device X.) An innocent pair could have produced at most six of the pirated content/keys. However, an innocent triplet picked at random could have produced all nine pirated content/keys, each member of the triplet having three content/keys in common with the guilty device. The number of such triplets are:

$$\binom{9}{3} * 60 * \binom{6}{3} * 60 * \binom{3}{3} * 60$$

Among all the $\binom{1,000,000,000}{3}$ triplets the probability that a triplet picked at random is in the above set is roughly 2 in 10^{18}. If the licensing agency is willing to assign apriori probabilities to the different numbers of attackers, and assuming that the attackers cannot deduce the code and therefore must act randomly, the license agency can perform a Bayesian analysis and conclude, based on the observed result, what the probability is that the indicated device X is, in fact, guilty.

One important caveat is traditionally addressed by defining the tracing problem to be finding a single member of the coalition, not finding the exact membership of the coalition. So the Bayesian analysis really reveals the probability that device X is an attacker, not that he is the sole attacker.

As one can see from the above simple example, without knowing the actual coalition size, the tracing has to be probabilistic. During tracing, every time a pirated copy is recovered, it increases the probability that the suspect device is actually guilty. That is the nature of the tracing when the actual coalition size is unknown. From the example above, we can also see during the process of figuring out the traitorous devices, it is possible to deduce the size of the active members in the coalition without having to assume the maximum coalition size. We believe performing probabilistic tracing and deducing the active coalition size much better fits the real world scenarios.

3.3 Disable Found Traitors and Continued Tracing?

Traditional traitor tracing is defined to be finding at least a traitor even if there may exist a coalition. They assume this traitor can be disconnected in some way. If there still is piracy, just repeat the same scheme. As we mentioned earlier, while it is sometimes possible to use legal means to disable traitors, it is not always possible to do that. In fact, technical means are oftentimes necessary in the lifetime of a traitor tracing system to help disabling traitors. In addition to this, we also challenge the assumption that tracing can simply be repeated with remaining traitors after disabling found traitors.

So, how does one technically disable a device that is found to be a traitor? For anonymous attack, we know each content is differently watermarked and encrypted. Each device is assigned a sequence of keys, one key for each content. To disable the traitorous device, one can render the compromised keys no longer usable for future content. But we know many devices might share a single compromised key. Therefore, revocation of a single key is impossible. We must revoke the unique set of keys assigned to a revoking device. Furthermore to make this work one must make sure that no two devices have many keys in common, so even if the system has been heavily attacked and a significant fraction of the keys in the system is compromised, all innocent devices will still have many keys that are not compromised.

For example, suppose each device is assigned a sequence of keys (called tracing keys) from a matrix, exactly one key per column. Each key enables the device to decrypt one content, and repeat from beginning when reaching the end of the column. The role of the sequence of keys assigned to each device in this system is similar to the media key discussed above in a broadcast encryption system. In order to disable a traitorous device, we will use Tracing Key Blocks (or TKB in short). TKB is generated by the license agency and distributed together with the new content in future. The purpose of the Tracing Key Block is to give all innocent devices a column they can use to calculate the correct key to decrypt the content, while at the same time preventing compromised devices (who have compromised keys in all columns) from getting to the correct answer. In an TKB there are actually many correct answers, one for each variation of the content. Let us simply call those answers output keys.

Now after some traitorous devices are disabled in Tracing Key Blocks, can we simply repeat the same tracing process as before for the remaining traitors? The answer is not trivial.

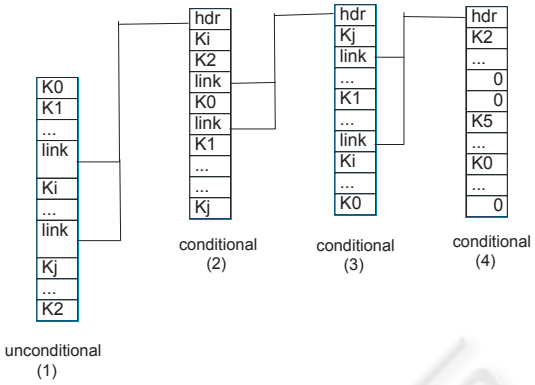

Figure 1: It is possible that the new TKB will not provide new tracing information for continued tracing

As we know, each column in a TKB contains an encryption of an output key in every un-compromised tracing key's cell. More precisely, in every un-compromised tracing key's cell in each column, it contains an encryption of one of the different correct output keys $D_{vi}(0 \leq i < k), k$ is the number of different correct answer in the system. Notice the same set of output keys are encrypted in each column, although they are distributed differently in different columns. For a particular output key D_{vi}, it can be obtained from any column in the TKB. A compliant (good) device will process TKB and obtain a correct output key from the first column in TKB that it has a non-revoked key. However, when there are still attackers in a coalition that have not been detected, the coalition can mix-and-match their revoked keys and non-revoked keys when processing TKB. In turn they have multiple ways to process TKB and get a valid output key to play back the content. They can choose in which column they want to use a non-revoked key to get a valid output key. It does not have to be in the first column. Moreover, different keys can be used in different columns to obtain the same variant D_{vi}. When the license agency observes a pirate copy corresponding to a particular output key D_{vi}, since it can be obtained from any column, the license agency has no way to exactly know which key has been used in obtaining that output key. The entire path that the undetected traitors goes through to process TKB can even look like from an innocent device or from a path that was never assigned to any device, thus untraceable.

The figure 1 illustrates the issue discussed here. Keep in mind that the output key has multiple valid versions. If the attackers combine the revoked keys with the keys that have not been detected, it is not always possible to know from which column the TKB processing ends to get a valid key.

To force the undetected traitors to reveal the keys they use when processing TKB, we must make sure

each column gets different variations so that when recovering a key/content-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 TKB. Each column only effectively gets q/c variations where c is the number of columns in the TKB. 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. See (Jin and Lotspiech, 2007) for more details.

As one can see, the challenge here is to make sure the newly released TKB can continue to provide tracing information to the license agency to enable continued tracing. Unfortunately this is not always possible. In fact, oftentimes it provides less or even no tracing information to the license agency for future traitor detection after the previous traitors are disabled. The simple assumption in traditional traitor tracing definition that tracing can simply repeated with same traceability does not hold.

In our opinion, when considering a complete cycle, a traitor tracing scheme contains the following three steps instead of the two steps defined in Section 2.

1. Assignment step: Assign versions of the content/key to currently known innocent devices
2. Forensic Analysis step: Based on the recovered forensic evidences (i.e., pirated content/keys), trace back to the traitors
3. Revocation step: loop to step 1 but exclude the currently discovered traitors, in other words, assign garbage to detected traitorous devices

The traceability of a complete traitor tracing system should be defined to be the traceability to detect all traitors in the system, including after revocation.

4 CONCLUSIONS

In this paper, we have examined what should be a good definition of a traitor tracing scheme that can lead to the design of efficient traitor tracing schemes. Traditionally traitor tracing system has been defined to find one traitor when assuming a maximum number of traitors in the system. We argue that definition is adequate and does not help one design efficient and practical traitor tracing system to use in real world. Keep in mind the ultimate goal of a traitor tracing scheme in real world is to detect and disable all active traitors. The traditional definition by detecting traitor one-by-one seems to be easy but actually does not help achieve this ultimate goal. Furthermore, assuming tracing can simply repeat after previously detected traitors are disabled in the system is wrong.

In our position, we believe a traitor tracing scheme should be defined to find all active traitors in the system and deduce the active coalition size. This includes the case that when some traitors are detected and disabled, the system should efficiently find the remaining traitors. The traceability is defined to detect all traitors in the system even with revocations of previously found traitors. Our new definition sets the correct tracing goal straight and could help leading to the design of an efficient and practical traitor tracing scheme.

REFERENCES

Chor, B., Fiat, A., and Naor, M. (1994). Tracing traitors. In *Crypto 1994, Lecture Notes in computer science*, volume 839, pages 480–491.

Fiat, A. and Naor, M. (1993). Broadcast encryption. In *Crypto 1993, Lecture Notes in computer science*, volume 773, pages 480–491.

J. N. Staddon, D. S. and Wei, R. (2001). Combinatorial properties of frameproof and traceability codes. In *IEEE Transactions on Information Theory*, volume 47, pages 1042–1049.

Jin, H. and Lotspiech, J. (2007). Renewable traitor tracing: a trace-revoke-trace system for anonymous attack. In *European Symposium on Research in Computer Security*.

Jin, H., Lotspiech, J., and Megiddo, N. (2008). Efficient coalition detection in traitor tracing. In *IFIP 23rd Internatinal Information Security Conference*.

Jin, H., Lotspiech, J., and Nusser, S. (2004). Traitor tracing for prerecorded and recordabe media. In *ACM workshop on Digital Rights Management*, pages 83–90.

Naor, D., Naor, M., and Lotspiech, J. B. (2001). Revocation and tracing schemes for stateless receivers. In *CRYPTO '01, Lecture Notes in Computer Science*, pages 41–62.

Safani-Naini, R. and Wang, Y. (2003). Sequential traitor tracing. In *IEEE Transactions on Information Theory*, volume 49, No.5, pages 1319–1326.