GENERALIZED TRAITOR TRACING FOR NESTED CODES

Avinash L. Varna∗
Dept. of ECE, University of Maryland
College Park, MD 20742

ABSTRACT
Nested or concatenated codes are often used for traitor tracing schemes as they require a small symbol size to accommodate a given number of users. Typically, tracing is performed on each layer of the code separately - tracing is first performed on the inner code and the information obtained is subsequently used to perform tracing on the outer code. In such situations, the collusion resistance is determined by the minimum of the coalition sizes that can be tolerated by the individual codes. Due to the small symbol size, the inner codes can tolerate a smaller number of colluders resulting in a small overall collusion resistance. Further, recovering more attacked versions does not enable identification of larger coalitions. To improve the collusion resistance, in this paper we propose to pass soft information from the inner code tracing to the outer code tracing. We demonstrate through simulations and formal analysis that the proposed technique improves the collusion resistance of tracing systems employing nested codes.

Index Terms—Traitor Tracing, Nested Codes, Multimedia Content Protection.

1. INTRODUCTION

The availability of high capacity storage media and easy access to high speed broadband networks has impacted the way multimedia is distributed and consumed. Consumers are demanding better quality content and the industry has moved to meet this demand. Movies are now distributed in high-definition - Digital Versatile Disks (HD-DVD) and Blu-Ray Disks (BD). Piracy of high definition content can lead to large losses in revenue for the entertainment industry. New technologies are in demand to prevent piracy and illegal redistribution of digital content. For example, a new content protection standard - the Advanced Access Content System (AACS) [1] was developed by a consortium consisting of content creators, hardware manufacturers and software companies to protect high definition DVDs.

A typical content protection system usually contains technologies that identify the guilty users (players) based on forensic analysis. Once the identity of a subverted player is known, a content protection system also provides technologies to revoke the identified subverted player and prevent further attacks using the same player. Different types of attacks are possible and oftentimes demand different types of forensic analysis techniques [2]. In this paper, we are particularly interested in anonymous attacks where the identity of the underlying subverted device is hidden, such as re-distribution of the movie in the clear or re-distribution of the movie encrypting keys. Traitor tracing tools [3] can be used to identify the devices used to launch such attacks.

Practical traitor tracing schemes often employ nested or concatenated codes. For example, AACS uses Sequence Keys as a traitor tracing mechanism [4, 5]. In the Sequence Key scheme, the key assignment to different players is based on a two-level nested Reed-Solomon code. The outer code corresponds to different versions of a movie that will be played by a given player and the inner code corresponds to variations within a movie that constitute a particular version. The nested code reduces the number of different versions of the same content needed to accommodate a given number of users, which corresponds to the symbol size of the inner code.

When an attacked movie is obtained, the detector attempts to determine which user or group of users is responsible for pirating the movie. A traitor detection algorithm based on set covering has been proposed for traitor tracing that requires fewer movies to detect an attacker [6] since it attempts to detect the entire coalition of users at once instead of detecting individual users. The set cover algorithm was proposed for a single layered code but can be used for nested codes by applying the algorithm on each level, to the inner code first and obtain the coalition of outer code symbols used to construct the attacked movie. These outer code symbols could then be used to apply set cover decoding on the outer code to identify the attackers.

Suppose we apply the set cover tracing on each level. For nested codes which have small inner code symbol size, the inner code is easily overwhelmed by a relatively small number of colluders, because with high probability, the attackers have every possible symbol amongst them. In this case, the set cover tracing algorithm applied on the inner code would not be able to correctly identify the outer code symbols involved in the attack which would lead to a complete decoding failure as the outer code now does not have any information to...
proceed. Thus, the set cover algorithm applied to each level separately may fail to identify attackers even when there is sufficient information to do so. It is highly desirable to develop techniques that can perform efficient tracing on nested codes since nested codes are commonly used for traitor tracking in real world.

In this work, we explore techniques to perform efficient tracing on nested codes, or more general multi-level codes by passing `soft‘ information to the outer level tracing. This partial information could be the number of segments each outer code symbol matches with or the locations of the segments each outer code symbol matches. By doing so, we are able to increase the collusion resistance of the traitor tracing system significantly.

2. SYSTEM DESCRIPTION

Our traitor tracing system consists of a nested / concatenated error correcting code. Nested codes are preferred in practice since they lead to small symbol sizes, which in turn translates into smaller number of required variations of the content. Specifically, consider a (inner) code $C_i$ of length $n_i$ over a field of size $q_i$ and an (outer) code $C_o$ of length $n_o$ over a field of size $q_o$. Let $|C_i| = q_o$. Then the code $C$ of length $n_i n_o$ obtained by mapping every symbol of $c \in C_o$ to a codeword in $C_i$ is a nested code.

Every user is assigned a codeword from the nested code $C$. In a typical implementation, the outer codeword would define which version of a particular file is given to the user. The set of the file assigned to the particular file version is defined by the inner code. Creation of different versions of a segment may be performed in different ways. Let us consider an example (also used in [4]). Each device is provided with a set of keys based on the codeword assigned to it. The inner code used is a $(15, 2)$ Reed-Solomon (RS) code over $GF(16)$. The outer code is chosen to be a $(255, 4)$ RS code over $GF(256)$. For a particular movie, certain segments critical to the plot of the movie are identified as segment key blocks. Each of these segments may then be differently watermarked to create the $q_i$ different variations. A player can decrypt and play only one out of these $q_i$ versions based on its codeword.

Every player plays back a unique variation of the different movies. Thus, if a single user pirates movies, and these are recovered, the guilty user can be identified easily. A coalition of attackers can launch a powerful attack by combining their different versions to create a pirated movie. We assume that for each segment, the attackers can only output a version which corresponds to at least one attacker’s version. This condition can be enforced by using robust watermarking techniques such as those in [7]. Upon obtaining a sequence of attacked movies, the detector attempts to catch the attackers using the set cover algorithm briefly described in the next section.

3. SET COVER ALGORITHM

The set cover algorithm proposed in [6], attempts to determine the entire coalition of attackers instead of an individual attacker. The intuition behind the set cover procedure is that it is easier to detect a coalition of users which can explain the observed sequence of movies rather than attempting to catch a single individual user. The reason for this is that even though there are exponentially many coalitions, the probability that an innocent coalition can correctly explain an observed sequence of attacked movies is negligibly small, given sufficient number of movies. In contrast, the probability that a single innocent user randomly matches a significant number of observed movies is quite high [6]. Hence, it is easier to identify the entire coalition at once rather than identifying each individual attacker.

Suppose we form $N$ codewords by randomly choosing symbols from an alphabet of size $q$ for each location. If we recover $M$ symbols, the expected number of innocent users whose codewords will match $x$ out of $M$ positions is given by $N \left( \frac{M}{x} \right) \frac{1}{q} \left( 1 - \frac{1}{q} \right)^{M-x}$. If we recover $M = 20$ for a code with symbol size $q = 256$ and $N = 10^9$, the expected number of users who match in 5 positions is approximately 13. In contrast, the expected number of random coalitions of size $T$ can completely explain the observed sequence of symbols is [6]

$$\left( \frac{N}{T} \right) \left( 1 - \left( 1 - \frac{1}{q} \right)^T \right)^m \approx \left( \frac{N}{T} \right) \left( \frac{T}{q} \right)^m,$$

which for $T = 4$ is approximately 0.6. Thus we see that the probability of false alarm is lower if we seek a coalition of size 4 which can explain the entire sequence of 20 symbols than if we attempt to find users who can explain 5 out of the 20 symbols.

The set cover algorithm iteratively determines the number and the members of the coalition as follows. Initialize the number of colluders $T = 1$. Check if any combination of $T$ users can explain the sequence of observed symbols. If not, increment $T$ and try to find a combination of $T$ users who can explain the observed symbols. Simultaneously, the algorithm keeps track of the probability that an innocent group of $T$ users will explain the observed sequence of symbols. The algorithm terminates when either it finds a cover, or the expected number of innocent coalitions which would cover the observed number of symbols (Eqn. (1)) is larger than a threshold. If the coalition cannot be determined with high confidence, then more symbols may be needed before an appropriate decision can be made.

For a nested code, we can use the algorithm on each level separately. The inner code decoding can be performed first to identify the outer code symbols of the attackers. This information could then be used for detection using the outer code. The problem with this approach is that, if the inner code decoding fails, then the outer code decoding fails too.
In our example, where the inner code is a (15, 2) code over 16 symbols, 5 codewords belonging to the inner code can always cover any attacked inner code sequence. If there are more than 5 colluders who each contribute equally on the inner code level, then the set cover algorithm on the inner level would report detection failure and no information is available at all for the outer code decoding. Hence, irrespective of the number of attacked movies observed, it would not be possible to catch coalitions of size larger than 5. This is intuitively dissatisfying as we would expect to be able to detect attacks by larger coalitions as the amount of available information increases. To remedy this problem, we propose to pass relevant partial information to the outer code decoding, to enable detection even in cases where set cover decoding on the inner level code would fail.

4. NESTED CODE SET COVER

As a first step, let us reformulate the set cover algorithm as an optimization problem. Suppose we observe a sequence of \( M \) symbols and we have \( N \) users. We construct a binary matrix \( A \) (of size \( M \times N \)) with each column corresponding to one user and each row corresponding to an observed symbol. The \((i, j)\)th entry of \( A \) is 1 if and only if the \(i\)th user’s \(j\)th symbol is the same as the \(j\)th observed symbol. Solving the set cover problem is then equivalent to solving the optimization problem:

\[
\hat{x} = \arg \min_{x_i \in \{0,1\}} \left[ \begin{array}{cccc} 1 & 1 & \ldots & 1 \end{array} \right]^T x
\]

\[
\text{subject to } A x \geq 1
\]

where by \( A x \geq 1 \) we denote the condition that each element of \( Ax \) is greater than or equal to 1.

For a nested code, a simple extension would be to apply the set cover algorithm on each individual level. Upon identifying the coalition of outer code symbols corresponding to each segment, we can treat them as we would other observed outer code symbols, and try to find a cover which can explain all the symbols.

As noted previously, this approach can fail if the inner code tracing cannot identify the coalition of outer code symbols. In such cases, we would like to pass some “soft” information to the outer code to enable tracing. Traitor tracing for shortened and corrupted fingerprints using soft information, namely, the probabilities of the individual symbols was considered in [8]. The general philosophy in our technique is the same. However, the information passed and the tracing algorithm used are different.

Suppose the inner code is of length \( n_i \) and has \( N_i \) codewords. This would imply that the outer code has symbol size \( N_i \). Suppose we observe \( M \) attacked movies corresponding to \( M \) outer code symbols. If the inner code detector is unable to uniquely determine the coalition of outer code symbols, it can just output the fraction of segments (corresponding to inner code symbols) each outer code symbol matches with. This “score” would be representative of the probability that the particular outer code symbol was part of the coalition that created the attacked movie.

Fig. 1. Number of times the entire coalition is successfully detected (100 trials)

The outer code detector then constructs a matrix \( A \) with \( M \) rows and column size equal to the number of users. The \((i, j)\)th entry in the matrix is the fraction of segments that user \( i \)’s \( j \)th inner codeword matches in the corresponding locations in the observed attacked sequence. The detector can then solve the set cover problem by solving the equivalent optimization problem (Eqn. 2). This optimization problem may not have a unique solution. However, the guilty coalition will be one of the minimum weight solutions. The detector can check if the solutions obtained explain all the observed symbols and hence eliminate spurious coalitions.

Another technique to avoid spurious solutions would be to output the locations where the inner code segments match, along with the number (fraction). Thus, for each outer code symbol, the inner code detector would output a binary vector of length \( n_i \) indicating whether the inner codeword corresponding to that outer code symbol matches with the observed symbols or not. The outer code detector then stacks these as columns to create the matrix \( A \) which now has \( M \times n_i \) rows. The \((i, j)\)th entry is 1 if the \(j\)th symbol of the nested codeword of user \( i \) matches the \(j\)th attacked symbol observed. The guilty coalition can then be found by solving Eqn. 2.

Solving Eqn. 2 would be NP hard. In practice, the tracing can be sped up by sorting the users in descending order according to their score and then trying to find a coalition that can explain the observed sequence, starting with the first. This would be faster since it is expected that at least some of the guilty users would have reasonably high scores. In our experiments, we found that when the size of the attacking coalition is not too large, the algorithm terminates within a few minutes with the solution.

4.1. Traceability of Our Proposed Technique

We test our proposed technique using a system with an inner RS code (15, 2) over \( GF(16) \) and an outer RS code (255, 3) over \( GF(256) \). Thus our system has \( 256^3 = 16 \times 10^6 \) users. The attackers randomly choose one of the available versions.
for each segment and concatenate them to form the attacked movie. For the tracing, we pass the score and location of the matching segments corresponding to each outer code symbol from the inner code tracing to the outer code level.

Fig. 1 compares the number of times the entire coalition is correctly detected from 30 attacked movies using the proposed algorithm and the set cover decoding applied individually to the inner and outer codes. The results presented are for 100 trials. From the figure, we observe that if the tracing is applied individually on the inner and outer codes, the system is unable to resist attacks by more than 5 colluders. This is due to the fact that any sequence of inner code symbols can be covered by 5 codewords. If the number of attackers is larger than 5, the inner code tracing is unable to identify the symbols used for the attack. Hence, irrespective of the number of attacked movies recovered, coalitions of more than 5 users cannot be identified. In contrast, using the proposed technique we can easily identify coalitions of 10 users from 30 movies. From Fig. 2 we see that the proposed technique outputs a unique coalition in all the trials. From these results, we can conclude that the proposed technique uniquely identifies the entire coalition. For larger coalitions, more attacked movies may need to be recovered before the coalition can be identified uniquely with high confidence.

We have performed a formal analysis to determine the number of movies that are required to identify a coalition with high confidence. Assuming that for each segment, the attackers output a segment randomly among their different versions, we calculate the probability that an innocent coalition will cover an observed sequence of movies. In this way the probability formulation is same as the one in [6]. Fig. 3 shows the number of movies needed to identify coalitions of attackers. The figure illustrates results for different inner code symbol size 8, 16 and 32, in each case creating 256 inner codewords. From the figure, we see that with approximately 30 movies, we can identify coalitions of up to approximately 12 attackers. Our simulation results are consistent with these theoretical results obtained.

5. CONCLUSIONS

In this paper, we have proposed a technique to improve the collusion resistance of tracing schemes employing nested codes by passing soft information from the inner code tracing to the outer code. Applying the tracing on each individual layer leads to a hard limit on the collusion resistance and obtaining more information in the form of attacked movies does not improve the maximum coalition size that can be resisted. In contrast, we showed through formal analysis and simulation that the proposed technique overcomes this hard limit. Larger coalitions of attackers can be uniquely identified by obtaining more information in the form of attacked movies and the collusion resistance of the proposed algorithm increases with the number of observed movies. As future work, we are looking to improve the traceability even more and also reduce the computational time for our proposed technique.

6. REFERENCES