Dynamic Traitor Tracing

Amos Fiat[1,2] and Tamir Tassa[2]

[1] Department of Computer Science, School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel
[2] Algorithmic Research Ltd., 10 Nevatim St., Petah Tikva, Israel

Abstract. Traitor tracing schemes were introduced so as to combat the typical piracy scenario whereby pirate decoders (or access control smartcards) are manufactured and sold by pirates to illegal subscribers. Those traitor tracing schemes, however, are ineffective for the currently less common scenario where a pirate publishes the periodical access control keys on the Internet or, alternatively, simply rebroadcasts the content via an independent pirate network. This new piracy scenario may become especially attractive (to pirates) in the context of broadband multicast over the Internet. In this paper we consider the consequences of this type of piracy and offer countermeasures. We introduce the concept of *dynamic* traitor tracing which is a practical and efficient tool to combat this type of piracy. We also consider the *static* watermarking problem, presented by Boneh and Shaw, and derive bounds on the performance parameters of the "natural majority algorithm".

Keywords. Broadcast, encryption, traitor tracing, watermarking, imprinting, pay TV, on-line algorithms.

1 Introduction

The subject of this paper is protecting ownership rights of intellectual property. The best example is that of pay TV systems where subscribers may access specific channels or programs by purchasing their viewing rights. In such systems, the content is distributed via terrestrial, cable or satellite broadcast and, hence, a *conditional access* system must be utilized in order to guarantee that only paying subscribers can access the content for which they have paid. But even though pay TV systems are the most outstanding realization of our model, there are others as well: conditional access systems are also used to protect pay services on The Web.

In this paper we address the issue of protecting ownership rights against piracy whereby unauthorized users get access to the content. Pirates make a business of breaking the security safeguards of the conditional access system and sell devices that allow unauthorized users to view the content illegally. To prevent such unauthorized access, cryptography is often used: the conditional access system makes use of secret keys in order to allow only legitimate users access to the content.

The use of tamper-resistant devices for conditional access systems is the norm, so as to prevent access to the underlying keys. However, recent advances in attacks on tamper resistant devices, most notably differential power analysis and timing attacks [6], have compromised unqualified reliance on tamper resistance. Thus, a more realistic model must assume that piracy will occur and, therefore, countermeasures should be taken once piracy has been observed. Such countermeasures should be capable of the following:

- Trace the source of piracy.
- Disconnect it and its dependent unauthorized users from further transmittal of information.
- Harming no legitimate users.
- Supply legal evidence of the pirate identity.

The traitor tracing schemes of Chor et al [2] adopt the following model: pirate decoders that allow access to the content may be manufactured but such decoders, if captured, must inherently contain identifying information that will allow the broadcaster to cut them off from future broadcasts. Additionally, the source of piracy can be detected and legal means can be taken.

To do so, Chor et al introduce a new form of cryptography that uses one encryption key and multiple distinct decryption keys, with the property that one cannot compute a new decryption key from a given set of keys. The traitor tracing schemes of [2, 8, 10] approximate such a scheme. Two cost measures are to be considered when implementing such schemes: storage requirements at the user end and the necessary increase of bandwidth.

The Achilles' heel of such traitor tracing schemes is their underlying assumption that pirates provide unauthorized subscribers with decoders capable of decoding the original broadcast. Such schemes would be ineffective if the pirate were simply to rebroadcast the original content using a pirate broadcast system.

This paper deals with the latter scenario: Even if the pirate rebroadcasts the original content to pirate users, countermeasures can be activated in order to trace and disconnect the so-called *traitors*, *i.e.*, the real subscribers controlled by the pirate.

To accomplish this, *watermarking methods* are implemented, allowing the broadcaster to generate different versions of the original content, with no noticeable degradation in the content quality. The schemes which we introduce and discuss here, use the watermarks found in the pirate copy to trace its supporting traitors. A fundamental assumption in this context is that it is possible to generate tamper-resistant watermarks that a pirate could not remove. Cox et al [3] have introduced methods to create such secure and robust watermarks.

Watermarking schemes were introduced and discussed by Boneh and Shaw in [1]. In their study they assumed that the content is watermarked once, prior to its broadcast. The schemes of [1] were designed to trace the source of piracy once a pirate copy of the content is captured. The traitor tracing schemes of [2] are similar in that sense: each decoder is personalized by a unique allocation of decryption keys, once, before it is sold to a subscriber. Only when a pirate decoder is captured, the traitor tracing schemes are activated in order to trace a legal decoder used in building the pirate unit. Both the watermarking schemes of [1] and the traitor tracing schemes of [2] are probabilistic. Namely, the evidence they provide against the suspected traitor is accompanied by a small error probability (that can be made as small as desired). It should be noted that even though the watermarking codes of [1] and the traitor tracing key assignment tools of [2] have, seemingly, an entirely different motivation, traitor tracing schemes can be translated into watermarking codes as described in [1].

Like [1, 2], we make use of marking codes but, unlike [1, 2], our codes are generated on the fly. In our model, we use the feedback from the pirate distribution network in order to lock onto the traitors much more efficiently. We refer to this latter model as *the dynamic model* while the former one [1, 2] is referred to as *the static model*. The dynamic model is very natural and has great practical applications in the context of protecting intellectual rights in broadcast systems. The static model, on the other hand, is suitable for electronic data distribution systems.

To understand the fundamental contribution of the dynamic model, we consider the following scenarios:

1. Dynamic schemes decide about the number of active traitors on the fly, based on the feedback from the pirate network, and adapt their behavior accordingly. That is impossible in the static model, where an a priori bound on the number of traitors is required (the lack of such a bound renders any static method completely unreliable).
2. Even if an a priori bound is known, but false incriminations of innocent users are strictly prohibited, there is an exponential performance improvement of dynamic methods over static ones. This exponential gap implies that static schemes are simply impossible in such settings.
3. If an a priori bound is known, and one allows a constant probability, $\varepsilon > 0$, of false incrimination, static schemes pay an additional $\log(1/\varepsilon)$ factor in performance that is not required by dynamic methods.

1.1 Organization of the Paper

The paper is organized as follows: in §2 we formalize the model, introduce the basic terminology and discuss relevant implementation issues. Then, in §2.3, we prove a fundamental result that connects the size of the marking alphabet to the number of active traitors. A byproduct of our analysis is that the probabilistic nature of the codes in [1] is inherent, *i.e.*, no code of that nature can avoid making errors.

§3 is devoted to the dynamic setting. Three deterministic algorithms are presented and compared: two of them have optimal spacial efficiency while the other one excels in temporal efficiency.

In §4 we discuss the static setting and study a scheme proposed in [2]. Using a more careful analysis, we are able to obtain performance estimates which are much better than those obtained in [2]. We note that this scheme, with our improved estimates, may be combined with the codes and distribution mechanism of [1] in order to yield schemes with optimal spacial efficiency.

Finally, in §5, we list several interesting open problems that our study raises.

1.2 Related Work

The concepts of frameproof codes and secure codes were defined in [1]. Additional explicit constructions of frameproof codes were given in [9].

There are a variety of slightly different definitions of frameproof and secure codes. Generally, a frameproof code is an assignment of codewords to users so that no coalition whose size is no more than some preset limit p can "frame" an innocent user. A coalition of size p can compute new codewords from the set of codewords assigned to its members. The rules by which new codewords can be computed vary slightly from paper to paper. The different rules refer specifically to what is permissible when combining two or more codewords to create another. For example, given two codewords x and y that differ in their ith coordinate, $x_i \neq y_i$, one can generate a new codeword z for which either

1. $z_i \in \{x_i, y_i\}$ (as in the *CFN*-model [2], which coincides with ours), or,
2. z_i is either an arbitrary element of the underlying alphabet or something entirely unrecognizable (as in the *BS*-model [1, 9]).

Rather than talk in terms of codewords, let us translate these two models to the watermarking terminology:

- Given two variants of a movie segment, v_1 and v_2, if the only possible choice for the pirate is to transmit either v_1 or v_2, then we are in the *CFN*-model.
- Given two variants of a movie segment, v_1 and v_2 ($v_1 \neq v_2$), if the pirate can produce any variant out of all possible variants, or something entirely unrecognizable, then we are in the *BS*-model [1, 9].

We justify our choice of model below, but first a few words to avoid confusion. We use the term "the *CFN*-model" somewhat misleadingly because [2] does not deal with the watermarking problem at all. Rather, [2] deals with the assignment of keys to decoders so as to recognize the source of a pirated decoder. One of the properties of cryptographic keys is that given two different symmetric keys, it (usually) makes little sense to try to combine them in some way and obtain a meaningful third key. Thus, if the pirate has to choose between using key k_1 or key k_2, he can choose either one of them, or none of them, but would not find it useful to use, say, $k_1 \oplus k_2$. In the translation between the traitor tracing schemes of [2] and watermarking schemes, the different keys are analogous to different variants of a segment, whence the term "the *CFN*-model" in the context of watermarking.

Given some variants of a movie segment, it would seem most unfeasible to compute a new valid variant. The reason for that is that in any reasonable watermarking scheme the pirate would not have the information essential to generate such a variant. It may be possible, however, to remove all watermarking information while paying the price of quality degradation. But even if that is possible, it would be difficult to do so, and the pirate would not necessarily know whether he was successful or not. This is why we find the *CFN*-model a more realistic model in this context. It should be noted that in our dynamic schemes, if we cannot recognize the variant that is currently transmitted by the pirate, we simply ignore the corresponding segment and wait for the next one. Even if the pirate is successful in removing the watermarking with probability q (the value of which is dictated by the technical difficulties, as well as by the need to have a rebroadcast with a reasonable quality), it implies a $1/(1-q)$ constant factor in convergence time.

Finally, from a practical perspective on the immediate future, we can justify our model for much the same reasons as in [2] (see §2.2).

In the static model, a related paper by Stinson and Wei [9] constructs frame-proof schemes as well as traceablity schemes. In this context, traceability schemes coincide with simple majority deterministic tracing algorithms that are not allowed to make any error. In [9, Theorem 5.5] they give a bound that connects all the parameters of the problem: the number of users, the size of the coalition of traitors, the size of the marking alphabet and the length of the codewords. That bound may be translated into a lower bound on the length of codewords which is proportional to the number of traitors times the log of the number of users. We conjecture in this paper that the true lower bound is much higher and is in fact exponential in the number of users.

Other related work about traitor tracing may be found in [4, 7, 8].

2 The Model

In our model, content consists of multiple *segments*, *e.g.*, a segment could be one minute worth of video. It is possible to generate multiple *variants* of each segment. Those variants must meet with the following two requirements:

- **Similarity.** Fundamentally, all variants carry the same information to the extent that humans cannot distinguish between them easily.
- **Robustness.** Given any set of variants, $v_1, \ldots, v_k$, it is impossible to generate another variant that cannot be traced back to one of the original variants, v_i, $1 \leq i \leq k$.

Clearly, those requirements place an upper bound on the number of variants that can be generated from a single content segment (the reader is referred to [3] where methods to generate such watermarks are introduced). Content for which some or all of the segments have been assigned variants is called a *watermarked content* or a *version*.

In general, the *watermarking problem* is to generate multiple versions of watermarked content so that, given a black market copy of that content, the watermarks embedded in that copy would lead to the identification of its source.

A *watermarking scheme* for tracing traitors consists of two essential parts:

1. **Watermark distribution:** An algorithm that assigns each subscriber a watermarked copy of the content.
2. **Tracing and incrimination:** An algorithm that, given an illegal copy of the content, uses the watermarks embedded in it in order to trace back at least one of the traitors that participated in producing that copy.

A watermarking scheme is called *deterministic* if it traces and incriminates all traitors and no one else but the traitors. On the other hand, schemes in which there is a small chance of false incrimination are referred to as *probabilistic*.

The two key performance parameters in this context are r, the number of different variants used per segment and m, the number of content segments. One way to view our model of watermarking is that it is an embedding of a codeword in the content, where r is simply the size of the marking alphabet and m is the length of the codeword.

The following terminology is used throughout the rest of the paper:

1. **The center** is the source of the content and its watermarked copies.
2. **The users**, or **subscribers**, denoted by $U = \{u_1, \ldots, u_n\}$, are recipients of the content.
3. Some of the users may collude in order to distribute illegal copies of the content to pirate subscribers. We refer to such users as **traitors** and to their coalition as **the pirate** and denote them by $T = \{t_1, \ldots, t_p\}$, $T \subset U$.
4. **The marking alphabet** that is used to generate codewords is denoted by $\Sigma = \{\sigma_1, \ldots, \sigma_r\}$.
5. For a given segment $1 \leq j \leq m$ and a mark $\sigma_k,\ 1 \leq k \leq r$, $S_k^j \subset U$ denotes the subset of subscribers that got variant σ_k of segment j.

We consider in this paper two settings: a dynamic setting and a static one.

The dynamic setting assumes on-line feedback from the pirate subscribers to the center. Such a scenario is feasible in cases like TV broadcast, where the pirate rebroadcasts the content, say, on the Internet. The center can therefore see the current pirate broadcast and adapt its watermark distribution in the next segments in order to trace the traitors efficiently.

In such a scenario, the number of variants that are transmitted simultaneously, r, is proportional to the bandwidth requirements, while m, the number of segments or search steps, is proportional to the time required to trace the traitors (the convergence time).

In **the static setting** there is a one time marking of the content per user. Only when a black market copy is found, the tracing and incrimination algorithm is activated. This model is suitable for, *e.g.*, DVD movie protection. Obviously, performance in such a rigid setting with no on-line feedback is less efficient than that in the dynamic setting. This setting is also somewhat less useful than the dynamic setting because there are fewer effective countermeasures: legal action post-factum is the only recourse (as opposed to the dynamic setting that allows immediate disconnection of the traitors).

As in the dynamic setting, r and m are relevant performance measures, but they have a slightly different significance. Here, r determines the relative extra expense required for watermarking, while m is limited by the maximal number of segments that can fit into the given content.

2.1 Control Overhead

A key issue is to control what users get what variant of every segment. The simplest way to do so is as follows:

1. Every user has a unique symmetric key in common with the center.
2. Prior to every segment transmission, the center distributes keys to users, using individually encrypted transmissions: If user i is to get variant ℓ of segment j, then the center sends an individually encrypted transmission to user i containing key K_ℓ^j, where all such keys are generated at random.
3. The center now transmits multiple variants of the jth segment, where variant ℓ is encrypted under key K_ℓ^j.

The broadcast overhead for implementing such a scheme is composed of two components:

1. Before each segment, the center needs to transmit individual (short) messages to every user that contain the relevant keys.
2. The center needs to broadcast multiple variants of every segment; this is a high overhead component because it multiplies the total bandwidth by the number of different variants.

There are a number of mechanisms that allow us to reduce this overhead. First, rather than using individually encrypted messages we can use broadcast encryption schemes [5]. At first glance it seems that this creates a problem because broadcast encryption schemes require an a priori knowledge of the number of traitors, whereas we claim that we do not need to know this. However, we never kill off a suspect user unless we know for sure that he is a traitor. Hence, we can start with an estimate on the number of traitors, and if this estimate turns out to be wrong, we can simply restart with a higher initial estimate for the broadcast encryption component.

Next, we do not necessarily have to change keys between segments for all users. In fact, we only need to change keys in case that a set of users is split up into two or more subsets, or if we perform a union between sets of users. Thus, even if one uses the naive approach (individual transmissions to every user) it turns out that our $2p+1$ algorithm, §3.3, only requires $O(np)$ individual transmissions for all segments.

However, the more expensive overhead is in the simultaneous transmission of multiple variants of a segment. Here, one can make use of the nature of the problem to reduce bandwidth overhead. Even if, say, 90% of the movie were transmitted entirely in the clear (and not watermarked), while only the remaining 10% were to be watermarked and protected, this would create problems for the pirate. A pirate copy that misses 10% of the movie is not very valuable. This means that we can transmit multiple variants for only a (relatively) small part of the movie, hence reducing the bandwidth overhead considerably.

2.2 Short-Term Practical Considerations

In the immediate future, it seems rather unlikely that the actual MPEG-II transmission will be rebroadcast over the Internet (due to lack of bandwidth). Thus, it may be that the setting described in this paper is not required in the immediate future. Hence, we briefly describe how to adapt our schemes for conditional access schemes used today.

All conditional access schemes today use rapidly changing symmetric keys to encrypt the content. These symmetric keys, known as "control words", are replaced (say, every 5 seconds) through the use of so-called "Entitlement Control Messages" (ECMs). An underlying hidden assumption in common to all these schemes is that the control words will not be retransmitted by the pirate to his subscribers.

This assumption is true if the bandwidth available to the pirate for retransmission is lower than that required to retransmit the control words. Thus, the center must set the control word change rate to reflect the bounds on the pirate transmission capabilities.

Nonetheless, the problem with this setting is that the pirate could still transmit the secret(s) used to obtain the control words from the ECMs.

Now, we can simply make use of dynamic traitor tracing schemes, where rather than watermarking multiple variants of the content, we encrypt the control words under several different keys (analogous to variants).

In this setting the control overhead is much lower (multiple ECM streams) and our model that disallows computation of a third variant from two existing variants is obviously justified.

2.3 Deterministic Lower Bound

Before discussing the dynamic and static settings separately, we state the following fundamental theorem which applies in both settings:

Theorem 1. *If the pirate controls p traitors then:*
(a) There exists a deterministic watermarking scheme with $|\Sigma| = p + 1$.
(b) No watermarking scheme that uses an alphabet of size $|\Sigma| \leq p$ can be deterministic.

In other words, a watermarking scheme must use an alphabet of size $p + 1$ at the least in order to trace and incriminate all traitors and no one but the traitors.

In the static setting, this requires to have an a priori bound on the number of traitors. However, as we shall see later, any deterministic scheme in this setting is bound to be impractical anyway and, therefore, only probabilistic schemes are considered.

In the dynamic setting, the scheme can learn on the fly what is the number of traitors and adapt its alphabet size accordingly; hence, no a priori information as to the number of traitors is required.

Proof. Here we prove part (b) of the theorem. As for the proof of part (a), see §3.2 and §3.4 where such schemes are described. Given some innocent subscriber of the system, $u \in U \setminus T$, we define $T_0 := T = \{t_1, \ldots, t_p\}$ and $T_i = T \cup \{u\} \setminus \{t_i\}$ for all $1 \leq i \leq p$. In addition, let us denote $T^* = T \cup \{u\}$. Now, assume that the pirate T adopts the following strategy: in segment j it rebroadcasts one of the variants σ_k for which $|S_k^j \cap T^*| \geq 2$. The existence of such a variant is guaranteed by the pigeon hole principle, since $|\Sigma| < |T^*|$. Clearly, the chosen subset S_k^j intersects T_i for all $0 \leq i \leq p$. Hence, it is impossible to distinguish between the real coalition of traitors, T_0, and the camouflage sets T_i, $1 \leq i \leq p$. Therefore, the scheme could never point out the p true traitors out of the $p+1$ subscribers in T^*.

We would like to point out that Theorem 1 is a generalization of [1, Theorem 4.2] which was restricted to the case $p = 2$. In addition, we proved this lower bound on the alphabet size under the more general assumption of robustness of the watermarks (whereas the proof of [1, Theorem 4.2] relied on the ability of the traitors to destroy marks).

3 The Dynamic Setting

3.1 Preliminaries

In the dynamic scenario, the pirate T broadcasts at every time segment j, $j \geq 1$, one of the variants owned by the traitors controlled by him, t_i, $1 \leq i \leq p$. Let us denote that variant by s_j and denote by B_j the pirate transmission up to time j, $B_j = (s_1, \ldots, s_j) \in \Sigma^j$ (those are available, say, by registering as a pirate user).

The goal of the watermarking scheme is to disconnect all subscribers in T, thus rendering the pirate inoperative. Additionally, it would be bad to disconnect innocent subscribers $u \in U \setminus T$. Hence, only deterministic schemes are considered in this case.

Formally, a dynamic watermarking scheme is a function $f : U \times \Sigma^* \mapsto \Sigma \cup \{\sigma_0\}$. For all $j \geq 1$, f induces a partition of U into the disjoint sets $S_k^j = \{u \in U \ : \ f(u, B_{j-1}) = \sigma_k\}$, $0 \leq k \leq r$. This is interpreted as follows:

1. At time $j \geq 1$, users $u \in S_k^j$, $k \geq 1$, get variant σ_k of content segment j.
2. At time $j \geq 1$, users $u \in S_0^j$ are disconnected, *i.e.*, get no variant of content segment j. We assume that $S_0^j \subset S_0^{j+1}$ for all $j \geq 1$, *i.e.*, disconnection is permanent.

In the following subsections we describe several deterministic schemes and study their performance in terms of r – the number of variants that they require in each segment, and m – the number of steps required to trace and disconnect all traitors. Those schemes do not require any a priori knowledge of p; instead, each of those schemes keeps track of a lower bound on the number of traitors. That value is initially set to zero and only when piracy is detected the scheme increases it to one. That lower bound is increased only when the findings of the scheme up to that point imply that this is valid. That lower bound is denoted by t in the first two schemes §3.2-3.3. In the third scheme §3.4, another related parameter appears and t has there a slightly different interpretation.

3.2 First Scheme: $r = p + 1$, impractical convergence time

The following straightforward scheme makes use of (no more than) $p+1$ variants in each segment. Therefore, it has an optimal spacial efficiency. However, its temporal efficiency is very bad as its convergence time is exponential in n.

1. *Set* $t = 0$.
2. *Repeat forever:*
   (a) *For all selections of* t *users out of* U, $\{w_1, \ldots, w_t\} \subset U$, *produce* $t+1$ *distinct variants of the current segment and transmit the* i*th variant to* w_i, $1 \le i \le t$, *and the* $(t+1)$*th variant to all other users, until the pirate transmits a recognized variant.*
   (b) *If the pirate ever transmits variant* i *for some* $i \le t$, *disconnect the single user* w_i *and decrement* t *by one. Otherwise, increment* t *by one.*

Clearly, this algorithm will trace and disconnect all traitors, t_i, $1 \le i \le p$, because when t reaches the value of p, one of the selections will be that in which $w_i = t_i$, $1 \le i \le p$; when that selection is made, either piracy stops or one of the traitors will incriminate himself. The convergence time for this algorithm, though, may be as large as $\binom{n}{p} + 2 \cdot \sum_{t=0}^{p-1} \binom{n}{t}$, hence it is impractical.

3.3 Second Scheme: $r = 2p + 1$, efficient convergence

Next, we present an algorithm that requires $2p+1$ keys but removes all traitors within $O(p \log n)$ steps. We note that any binary decision tree for determining all p traitors within a user group of size n has a depth of $p \log n$, as implied from the information theoretic bound.

Throughout this algorithm, the set of subscribers, U, is partitioned into $2t+1$ subsets, $U = \cup_{S \in P} S$, where $P = \{L_1, R_1, \ldots, L_t, R_t,\ I\}$, and each of those sets receives a unique variant. Hence, there are never more than $2t+1$ simultaneous variants; since t – the lower bound on the number of traitors – never exceeds p – the true number of traitors, the upper bound on the size of the alphabet, $|\Sigma| \le 2p+1$, is respected.

An invariant of the algorithm is that the union $L_i \cup R_i$ contains at least one traitor for all $1 \le i \le t$. I is the complementary subset of users that is not known to include a traitor.

1. *Set $t = 0$, $I = U$, $P = \{I\}$.*
2. *Repeat forever:*
   (a) *Transmit a different variant for every non-empty set of users $S \in P$.*
   (b) *If the pirate transmits a variant v of the current segment then:*
      – *If v is associated with I, increment t by one, split I into two equal sized subsets, L_t and R_t, add those sets to P and set $I = \emptyset$.*
      – *If v is associated with one of the sets L_i, $1 \leq i \leq t$, do as follows:*
         i. *Add the elements in R_i to the set I.*
         ii. *If L_i is a singleton set, disconnect the single traitor in L_i from the user set U, decrement t by one, remove R_i and L_i from P and renumber the remaining R_i and L_i sets in P.*
         iii. *Otherwise (L_i is not a singleton set), split L_i into two equal sized sets, giving new sets L_i and R_i.*
      – *If v is associated with one of the sets R_i, $1 \leq i \leq t$, do as above while switching the roles of R_i and L_i.*

Theorem 2. *The watermarking scheme which the above algorithm implements, traces all p traitors within $m = p \log n + p$ time steps, while using no more than $r = 2p + 1$ simultaneous variants.*

Proof. It is clear that at any given stage, the union $L_i \cup R_i$, $1 \leq i \leq t$, contains at least one traitor (this is an invariant of the algorithm). Hence, the number of $\{L_i, R_i\}$ pairs, t, cannot exceed the total number of traitors, p. Since the scheme uses at each stage no more than $2t + 1$ variants, the upper bound of $2p + 1$ simultaneous variants is respected.

As for the convergence time, consider a sequence of tracing steps through which a traitor is isolated in successively smaller subsets, L_i or R_i. Clearly, each single traitor will be isolated within $\log n$ steps. Hence, all traitors will be isolated within $p \log n$ steps. Once all traitors are isolated, the pirate's broadcast must incriminate them all after additional p steps.

3.4 Third Scheme: $r = p + 1$, improved convergence time

Here, we present another algorithm that uses an optimal alphabet of size $p+1$. Its convergence time is bounded by $O(3^p p \log n)$ which is a dramatic improvement over $\binom{n}{p} \approx n^p$ of the scheme in §3.2, though still non-polynomial in p. Our new algorithm is very similar to the previous one, §3.3, in the sense that the partitions that it uses are of the same form, $U = \cup_{S \in P} S$, $P = \{L_1, R_1, \ldots, L_t, R_t, \; I\}$, and it has the same invariants: the union $L_i \cup R_i$ contains at least one traitor for all $1 \leq i \leq t$, while I is not known to include any traitor.

The difference between the two algorithms (which is manifested most notably in their running time) stems from the fact that we may not have sufficient variants for all the $2t+1$ sets in P (due to the tighter restriction on the simultaneous number of variants). Hence, if in the previous algorithm we had only one dynamic parameter, t, that indicated both the number of $\{L_i, R_i\}$ pairs and the current lower bound on the number of traitors, in this algorithm there are two dynamic parameters:

1. k, the current lower bound on the number of traitors (*i.e.*, how many traitors are known to exist at this stage of the search), and
2. t, the number of pairs of subsets, $\{L_i, R_i\}$, in the partition P of U. Each of those pairs is known to include at least one traitor.

Clearly, $k \geq t$; later, we shall see that $k \leq 2t$. Hence, the knowledge that the tracing scheme holds in each step may be summarized as follows:

$$|T| \geq k \quad \text{and} \quad |(L_i \cup R_i) \cap T| \geq 1 \quad 1 \leq i \leq t \ .$$

In the $2p+1$ algorithm, having the luxury of assigning a unique variant to each set $S \in P$, we were guaranteed to make a progress in every step, where a progress means the splitting of one of the sets in P towards closing on the traitor(s) in that set. Here, however, we cannot do so since we are limited to use no more than $r = k+1$ different variants in each step. Hence, instead of achieving a progress in each step, the algorithm that we present below is guaranteed to achieve a progress within a finite number of steps.

1. *Set $t = 0$, $k = 0$, $I = U$, $P = \{I\}$.*
2. *Repeat forever:*
   (a) *For every selection of $\{S_1, \ldots, S_k\} \subset P$ where $S_i \in \{L_i, R_i\}$, $1 \leq i \leq t$, and S_i, $t+1 \leq i \leq k$ are any other $k-t$ sets from P, produce $k+1$ variants, σ_i, $1 \leq i \leq k+1$. Transmit σ_i to S_i for all $1 \leq i \leq k$, while all remaining users get variant σ_{k+1}.*
   (b) *Assume that the pirate transmits at some step a variant σ_i that corresponds to a single set in P (when $k < 2t$ those are the variants σ_i where $1 \leq i \leq k$; when $k = 2t$, on the other hand, all variants correspond to a single set, since then also σ_{k+1} is transmitted to just one set). In that case:*
      i. *If σ_i corresponds to an L_i set, then that set must contain a traitor. In that case we add the corresponding complementary set, R_i, to I and split L_i into two equal sized sets giving a new $\{L_i, R_i\}$ pair. In this case neither t nor k changes but eventually, when the size of the incriminated set is one, we may disconnect the traitor in that set. When this happens, we restart the loop after decrementing k and t by one.*
      ii. *If σ_i corresponds to an R_i set, we act similarly.*
      iii. *If σ_i corresponds to I, it allows us to increment t by one (and k as well, if k was equal to t), split I into a new $\{L_t, R_t\}$ pair, set $I = \emptyset$ and restart the loop.*
   (c) *If $k < 2t$ and the pirate always transmits σ_{k+1}, then after completing the entire loop we may increment k by one and then restart the loop.*

Given k and t, the basic loop consists of $2^{2t-k+1}\binom{t}{k-t-1} + 2^{2t-k}\binom{t}{k-t}$ rounds. Since, in the worst case, we may need to repeat the loop from $k = t$ to $k = 2t$ until we split a set, we are guaranteed to make a progress in the form of splitting a set within no more than $2 \cdot 3^t - 1$ steps (which equals the sum over k of the above terms). This is always bounded by $2\cdot 3^p$. Hence, convergence is guaranteed within no more than $(2\cdot 3^p p\log n+p)$ steps. This bound is not tight, but, on the other hand, it is clear that any upper bound on the convergence time cannot be less than $O(2^p p\log n)$ rounds. Hence, this algorithm is exponential in p.

Note that this algorithm actually combines the two previous ones. It uses the same search tree as the $2p+1$ algorithm of §3.3. However, when a gap is created between k and t, the previous $p+1$ algorithm of §3.2 is implemented in order to trace the additional $k-t$ subsets of P that contain a traitor. We could, of-course, avoid the inefficient algorithm of §3.2 and, instead, implement again the algorithm of §3.3 in a recursive manner. However, that would make the algorithm quite intricate, while not improving its convergence time substantially.

4 The Static Setting

4.1 Preliminaries

In this scenario, the content is marked once, by dividing it into m segments and marking each segment using the marking alphabet Σ, $|\Sigma|=r$. Each user is given a unique watermarked copy (version) of the content that is randomly chosen out of the r^m possible versions. The following notations and assumptions are used throughout this section:

- $C(u)$ is the copy, or version, that user $u\in U$ got. Namely, C is a one-to-one random function from U to Σ^m, which is assumed to be uniformly distributed.
- $C(T)$ is the closure of $\{C(t_1),...,C(t_p)\}$; *i.e.*, all vectors $C\in\Sigma^m$ for which $C_j\in\{C(t)_j :\ t\in T\}$ for all $j=1,...,m$. In other words, $C(T)$ consists of all versions that can be produced by T from the p versions of its members.
- C^* stands for the copy that T produced and distributed in the black market.
- For any $x,y\in\Sigma^m$, $M(x,y)=\frac{1}{m}\cdot|\{j: x_j=y_j,\quad 1\le j\le m\}|$ denotes their *matching score*.

We consider the natural majority algorithm: given a black market copy, C^*, the tracer incriminates and punishes that user u who has a maximal *matching score*, $M(C(u),C^*)$. This is the same algorithm that was suggested and studied in [2]. We carry out here a more careful analysis that enables us to improve the performance estimates that were obtained in [2].

Note that this scheme is the static analogue to the dynamic schemes described in the previous section, because all we care about are exact "matches" and the information on the matches is taken in each segment separately (as opposed to, *e.g.*, the Boneh-Shaw scheme [1] where the matching information is considered in the context of groups of segments).

The best strategy for the pirate against the majority algorithm is to select that codeword $C^*\in C(T)$ that minimizes $\max_{t\in T} M(C(t),C^*)$. However, finding the exact solution for this problem seems to be a difficult task. A random choice of C^* seems like a more practical solution for the pirate. We will assume that

C^* is uniformly distributed amongst all vectors in $C(T)$, *i.e.*, for each segment j, one of the available variants, $\{C(t)_j : t \in T\}$, is chosen uniformly at random (without repetitions).

Summary of results. Hereinafter we assume that U, T and Σ are given (where $n > p \geq 1$ and $r \geq 2$), as well as a small parameter $\varepsilon > 0$. We set the probability space to be that which consists of all possible tracer's allocations of codewords, $C(\cdot)$, and pirate's selections of $C^* \in C(T)$ with uniform distributions. The question which arises is as follows: Can we find a lower bound M such that the above described scheme, when using $m > M$ segments, would incriminate a true traitor in probability of at least $1 - \varepsilon$? In Theorem 3 we provide a positive answer to this question.

Next, we provide a simple algorithm to compute a lower bound M_1 such that the scheme would perform as desired for all $m \geq M_1$. This lower bound is significantly better than the lower bound obtained in [2].

We also study the asymptotic behavior of the lower bound for m in the two regimes $r \geq p + 1$ and $r < p + 1$. We note that also here, like in the dynamic setting, the value $r = p + 1$ is a special value that plays a significant role.

Deterministic methods vs. probabilistic ones. In the dynamic setting we concentrated on deterministic algorithms. In the static setting, however, not having the freedom of deciding on the allocation of variants in the next segment based on the findings in the previous segments, we restrict our discussion to probabilistic algorithms. This is because we have been unable to find a static deterministic algorithm that is more efficient than the exponential scheme in §3.2. One obvious open problem is proving that subexponential deterministic static schemes are impossible, or the converse.

Note that by relaxing the deterministic incrimination demand to a probabilistic one, we are able to break the barrier of $r = p + 1$ and achieve such a relaxed incrimination even with the minimal size of marking alphabet, $r = 2$, though, as shown later, with a very high price in m.

4.2 Lower Bounds

Under the assumptions and definitions given in §4.1, the following holds:

Lemma 1. *There exists a constant $b > 1$, depending solely on p and r, such that for any $t \in T$, $M(C(t), C^*) \underset{m\to\infty}{\longrightarrow} \frac{b}{r}$ a.e.*

Proof. For every $u \in U$, let $x(u)_j$, $1 \leq j \leq m$, be the following indicator random variables:

$$x(u)_j = \begin{cases} 1 & \text{if } C(u)_j = C^*_j \\ 0 & \text{otherwise} \end{cases} \quad . \tag{1}$$

We also define ξ_p^r to be the random variable that counts the number of different letters in a word of length p over an alphabet of size r and we let

$$\alpha_{p,r}(k) = Pr[\xi_p^r = k] \ . \tag{2}$$

With this, we get that for every $t \in T$,

$$E[x(t)_j] = Pr[x(t)_j = 1] = \tag{3}$$

$$\sum_{k=1}^{r} Pr\left[C(t)_j = C_j^* \;\middle|\; |\{C(t)_j :\ t \in T\}| = k\right] \cdot \alpha_{p,r}(k) = \sum_{k=1}^{r} \frac{\alpha_{p,r}(k)}{k} \ . \tag{4}$$

Since $\alpha_{p,r}(k) > 0$ for all $1 \le k \le r$, it follows that

$$E[x(t)_j] = \frac{b}{r} \tag{5}$$

for some $b > 1$ that depends solely on p and r. Finally, since $x(t)_j$ are independent and have the same distribution, and $M(C(t), C^*) = \frac{1}{m} \cdot \sum_{j=1}^{m} x(t)_j$, then, by The Strong Law of Large Numbers, it converges to its expected value a.e.

Theorem 3. *There exists m sufficiently large for which*

$$Pr\left[\ M(C(u), C^*) < \max_{t \in T} M(C(t), C^*) \quad \forall u \in U \setminus T\ \right] > 1 - \varepsilon. \tag{6}$$

Proof. In view of Lemma 1, we may approximate the maximal score of the traitors by

$$\mu := \frac{b}{r} \ . \tag{7}$$

Hence, we aim at showing that, for sufficiently large m,

$$Pr\,[\ M(C(u), C^*) < \mu \quad \forall u \in U \setminus T\ ] > 1 - \varepsilon \ . \tag{8}$$

Fixing $u \in U \setminus T$ and denoting $a = Pr\,[\ M(C(u), C^*) \ge \mu\ ]$, inequality (8) is equivalent to $(1-a)^{n-p} > 1 - \varepsilon$, and this inequality holds if

$$a < \frac{\varepsilon}{n} \ . \tag{9}$$

Hence, we aim at finding m for which (9) holds. To this end, we observe that

$$a = Pr\left[\ \frac{1}{m} \sum_{j=1}^{m} x(u)_j \ge \mu\ \right] \tag{10}$$

where $x(u)_j$ were defined in (1). Since $x(u)_j \sim B(\frac{1}{r})$, $1 \le j \le m$, and are independent, we may apply the normal approximation and conclude that

$$X := \frac{r \cdot \sum_{j=1}^{m} x(u)_j - m}{\sqrt{m(r-1)}} \sim N(0,1) \tag{11}$$

for sufficiently large m. Hence, in view of (7), (10) and (11), the inequality in (9) takes the form $Pr\left[\ X \ge \frac{b-1}{\sqrt{r-1}} \sqrt{m}\ \right] < \frac{\varepsilon}{n}$, and that holds if

$$m > M_1 := \frac{r-1}{(b-1)^2} \cdot \left(\mathrm{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \qquad \text{where } \ \mathrm{erfc}(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} dt \ . \tag{12}$$

The proof of Theorem 3 is constructive, in the sense that it provides an expression for the lower bound M_1. That lower bound involves the constant b which may not be computed explicitly. However, by (4) and (5),

$$b = r \cdot \sum_{k=1}^{r} \frac{\alpha_{p,r}(k)}{k} \ , \tag{13}$$

and that may be easily evaluated using the following self-explanatory formulae for computing $\alpha_{p,r}(k)$, (2):

$$\alpha_{1,r}(k) = \begin{cases} 1 & \text{if } \ k = 1 \\ 0 & \text{if } \ k \neq 1 \end{cases} \quad ; \quad \alpha_{p,r}(0) = 0 \ \ ; \tag{14}$$

$$\alpha_{p+1,r}(k) = \frac{k}{r} \cdot \alpha_{p,r}(k) + \left(1 - \frac{k-1}{r}\right) \cdot \alpha_{p,r}(k-1) \ , \tag{15}$$

where $p, r \geq 1$ and $1 \leq k \leq r$.

The case $r \geq p+1$. It is always safe to say that $\max_{t \in T} M(C(t), C^*) \geq \frac{1}{p}$. Hence, in this case we may take in (7) $b = \frac{r}{p}$ instead of (13). Substituting this explicit value for b in (12), we arrive at the following lower bound for m:

$$m > M_2 := \frac{(r-1)p^2}{(r-p)^2} \cdot \left(\operatorname{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \ . \tag{16}$$

Clearly, the lower bound in (16), M_2, is worse than that in (12), M_1. Let us consider the asymptotic behavior of M_2 for large values of p and see how it compares to that of M_1. It turns out that the answer to these questions depends on the relation between r and p:

• If $r = \gamma \cdot p$ where $\gamma = \text{Const} > 1$, then, by (16),

$$M_2 = \frac{\gamma \cdot p - 1}{(\gamma - 1)^2} \cdot \left(\operatorname{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \sim \text{Const} \cdot p \cdot \left(\operatorname{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \ .$$

The lower bound that was obtained in [2] under the assumption $r = 4p$ was $M_{\text{CFN}} = \frac{4}{3} p \log_2 \left(\frac{n}{\varepsilon}\right)$. Comparing the three bounds in the case $r = 4p$, we find that $M_1 < M_2 < M_{\text{CFN}}$ while $\frac{M_2}{M_1} \approx 1.377$ and $\frac{M_{\text{CFN}}}{M_1} \approx 3.629$. Therefore, in practice, M_1 should be used.

• If $r = p + \gamma$ where $\gamma =$Const, then, by (16),

$$M_2 = \frac{(p + \gamma - 1) \cdot p^2}{\gamma^2} \cdot \left(\operatorname{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \sim \text{Const} \cdot p^3 \cdot \left(\operatorname{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2 \ .$$

However, as numerical computations indicate, M_2 becomes meaningless in this case and M_1 should be evaluated in order to get a reasonable estimate for the required lower bound on m.

The case $r \leq p$. The majority scheme works also when $r \leq p$, even with the minimal r, *i.e.*, $r = 2$. However, the lower bound M_1 increases dramatically and the scheme becomes impractical; for example, when $r = 2$, $M_1 = 4^{p-1} \cdot \left(\mathrm{erfc}^{-1}(\frac{\varepsilon}{n})\right)^2$. The reason for that is that the smaller r is, the better are the chances that the traitors would have in each segment a full or almost full set of variants. In other words, the smaller r is, the closer $C(T)$ is to Σ^m and then the traitors could produce almost any version. For instance, if $p > r \ln r$ then, by the coupon collector problem, there are good chances that in a given segment j, the traitors have the complete set of variants, $\{C(t)_j : t \in T\} = \Sigma$. However, those segments in which the pirate lacks some variants, enable the scheme to relate a given black market copy, C^*, to T, provided that there are enough of those segments, *i.e.*, provided that m is sufficiently large. Hence, in applications where the size of of the watermarking alphabet is strictly restricted, more effective methods should be applied.

5 Open Problems

It is important to understand the underlying performance considerations which one needs to consider: bandwidth, storage and computation time. Some of the published results on various broadcast problems are seemingly irrelevant because they do not deal with the performance characteristics of the solution. One important task is to give a unified analysis of the various solutions proposed in the literature.

As for the present study, the open problems that it raises are as follows:

1. Devising a probabilistic algorithm in the dynamic model. There are two settings to consider in this context: (a) known allocation of codewords (the pirate knows the codewords of all users and not just of those he controls), and (b) oblivious allocation of codewords.
2. Finding a deterministic dynamic algorithm based on a minimal alphabet, $r = p + 1$, with a convergence time that is polynomial in p.
3. Proving or disproving that any deterministic static scheme is exponential (in the number of segments m).

Acknowledgments

The authors gratefully acknowledge interesting and valuable conversations with Omer Berkman, Jacob Goldberger and Benny Pinkas.

References

1. D. Boneh and J. Shaw, Collusion-Secure Fingerprinting for Digital Data, *IEEE Transactions on Information Theory*, vol. 44, no. 5 (1998), pp. 1897–1905 (see also Proc. Crypto 95, Springer LNCS 963, 1995, pp. 452-465).

2. B. Chor, A. Fiat, and M. Naor, Tracing Traitors, *Proc. Crypto 94*, Springer LNCS 839 (1994), pp. 257–270.
3. I.J. Cox, J. Kilian, T. Leighton and T. Shamoon, A Secure, Robust Watermark for Multimedia, *Information Hiding*, Springer LNCS 174 (1996), pp. 185–226.
4. C. Dwork, J. Lotspiech and M. Naor, Digital signets: Self-Enforcing Protection of Digital Information, *28th Symposium on the Theory of Computation* (1996), pp. 489-498.
5. A. Fiat and M. Naor, Broadcast Encryption, *Proc. Crypto 93*, Springer LNCS 773 (1993), pp. 480–491.
6. Paul Kocher, Cryptography Research, *http://www.cryptography.com/dpa/index.html*
7. M. Naor and B. Pinkas, Threshold Traitor Tracing, *Proc. Crypto 98*, Springer LNCS 1462 (1998), pp. 502-517.
8. B. Pfitzmann, Trials of Traced Traitors, *Information Hiding*, Springer LNCS 1174 (1996), pp. 49–64.
9. D. R. Stinson and R. Wei, Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes, *SIAM Journal on Discrete Mathematics*, vol. 11, no. 1 (1998), pp. 41-53.
10. J. Schwenk, J. Ueberberg, Tracing Traitors using Finite Geometries, *manuscript.*