J. Cryptology (2005) 18: 167–183
DOI: 10.1007/s00145-004-0214-z

Journal of
CRYPTOLOGY

Low Bandwidth Dynamic Traitor Tracing Schemes

Tamir Tassa
Division of Computer Science,
The Open University,
Ràanana, Israel
tamirta@openu.ac.il

Communicated by Dan Boneh

Received 7 June 2002 and revised 20 June 2004
Online publication 21 October 2004

Abstract. Dynamic traitor tracing schemes were introduced by Fiat and Tassa in order to combat piracy in active broadcast scenarios. In such settings the data provider supplies access control keys to its legal customers on a periodical basis. A number of users may collude in order to publish those keys via the Internet or any other network. Dynamic traitor tracing schemes rely on the feedback from the pirate network in order to modify their key allocation until they are able either to incriminate and disconnect all traitors or force them to stop their illegal activity. Those schemes are deterministic in the sense that incrimination is always certain. As such deterministic schemes must multiply the critical data by at least $p+1$, where p is the number of traitors, they may impose a too large toll on bandwidth. We suggest here probabilistic schemes that enable one to trace all traitors with almost certainty, where the critical data is multiplied by two, regardless of the number of traitors. These techniques are obtained by combining dynamic traitor tracing schemes with binary fingerprinting techniques, such as those proposed by Boneh and Shaw.

Key words. Broadcast encryption, Traitor tracing, Fingerprinting, Watermarking, Binary codes, Pay-TV, On-line algorithms.

1. Introduction

The concept of Dynamic Traitor Tracing was introduced in [7] in order to combat piracy in dynamic data distribution systems. This was a natural extension of the concepts of *Traitor Tracing*, *Watermarking* and *Fingerprinting* that were relevant for static data distribution systems. A data distribution system is any setting in which a data provider (the *center*) is providing data to a large group of paying users. Piracy occurs when one or few legal users redistribute the data for which they paid to illegal customers, thus rendering financial losses to the legitimate data provider. Hence, tracing techniques are required in order to find the source of information leakage, disconnect it and press charges against those traitorous users.

In order to distinguish between the static and dynamic models, we refer henceforth to the legal data distribution, from the center to the paying users, as the *primary* data distribution, while the illegal data distribution from the traitors to the illegal customers is referred to as the *secondary* data distribution.

A *static* data distribution system is one in which the primary data distribution completely precedes in time the secondary data distribution. Namely, primary and secondary data distributions are perceived as actions that have no duration: the first occurs at some time t_1 and the second occurs at a later time $t_2 > t_1$. Assume that the data provider captures such secondary distributed material. Obviously, if all legal users get the very same data, there is no way of tracing back one of the traitors, given the pirate material. Hence, it is necessary to personalize the data prior to primary distribution in such a way that any captured pirate material would enable the tracing of at least one of the colluding traitors. Traitor tracing schemes, watermarking, fingerprinting, traceability schemes and parent-identifying codes are different terms that were given to techniques that were designed to answer this need. One of the reasons for the plurality of terms is the diverse scenarios that give rise to this problem. In the context of decoders for Pay-TV, one usually thinks in terms of "traitor tracing". On the other hand, schemes that are designed to protect visual material are usually called "watermarking" or "fingerprinting". "Traceability schemes" or "parent-identifying codes" are the mathematical tools that such schemes use.

In view of the above, we feel that it is imperative to set a general framework that encompasses all scenarios of data distribution and all counter-piracy methods.

The Traitor Tracing Framework. Let $U = \{u_1, \ldots, u_n\}$ denote the set of users and $T = \{t_1, \ldots, t_p\} \subset U$ be the subset of traitorous users. T is called *the pirate* while its members are referred to as *traitors*. An algorithm that aims at locating the traitors is called a traitor tracing scheme. Such a scheme consists of the following ingredients:

- A marking alphabet, $\Sigma = \{\sigma_1, \ldots, \sigma_r\}$.
- A codebook Γ that is used for personalizing copies of the data prior to its distribution. Γ is an (r, ℓ, n)-code, meaning that $\Gamma = \{w^1, \ldots, w^n\} \subset \Sigma^\ell$, $|\Sigma| = r$.
- A personalization scheme $P\colon U \to \Gamma$ that determines how to mark the data that is provided to a particular user with a codeword from the codebook.
- A generation assumption: let $P(T) = \{P(t_1), \ldots, P(t_p)\}$ denote the set of codewords in the copies that are owned by the traitors. We let $\langle P(T)\rangle$ denote the set of codewords that could be generated by the pirate and be placed in the pirate copy. $\langle P(T)\rangle \subset (\Sigma \cup \{?\})^\ell$, where "?" denotes an unreadable mark. Different assumptions were made in different studies about the strength of the generation operation $\langle\cdot\rangle$, depending on the underlying application.
- A tracing algorithm that, given a pirate copy, aims at tracing back (at least) one traitor that collaborated in producing that copy. This algorithm may be therefore viewed as a function $A\colon \langle P(T)\rangle \to U$. Obviously, a desired property is that whenever $w \in \langle P(T)\rangle$, $A(w) \in T$. However, this is not always the case, as indicated by the next ingredient.
- An upper bound on the probability of false incrimination, $\varepsilon \geq 0$. In other words, if $w \in \langle P(T)\rangle$, then $\text{Prob}(A(w) \in U\backslash T) \leq \varepsilon$. Traitor tracing or watermarking schemes in which $\varepsilon = 0$ are called *deterministic* while those in which false incrimination may occur are called *probabilistic*.

We exemplify the above framework with two fundamental examples.

Example 1: Pay-TV Decoders. Chor et al. [4] considered the setting of a Pay-TV system, where Conditional Access techniques are implemented in order to deny access to content from non-paying users. In order to achieve that goal, the stream of content that is broadcast from the center is encrypted, and each paying user is given a decoder with embedded keys that are capable of decrypting this transmission. As such decoders (typically, smartcards) are merely tamper-resistant, and not tamper-proof [8], highly motivated and capable users may read the secret keys from their own decoder in order to manufacture illegal pirate decoders and start their own business. Hence, in order for the center to be able to trace the source of such piracy, the personal keys are marked in some manner. The following is a basic technique of marking keys. The center determines two parameters—r (the size of the marking alphabet) and ℓ (the length of codewords)—in a manner that will be explained later. It then generates $r \cdot \ell$ random and independent encryption keys, $(k_{i,j})$, $1 \le i \le r$, $1 \le j \le \ell$. Next, each personal decoder is provided with a unique selection of ℓ of those keys—one from each column. Namely, denoting $\Sigma = \{1, \ldots, r\}$, the center employs a personalization function $P\colon U \to \Gamma \subset \Sigma^\ell$ and then each user $u \in U$ gets the following personal key:

$$k(u) = \{k_{i(j),j} \ : \ 1 \le j \le \ell \ , \ i(j) = P(u)_j\}.$$

Let s be the secret that needs to be communicated to all paying users (e.g., s is the session key to be used throughout the next hour). Then s is broken in a random manner into ℓ parts, $s = s_1 \oplus \cdots \oplus s_\ell$, where $\oplus$ denotes XOR. Then the secret s_j is encrypted r times with respect to each of the keys $k_{i,j}$, $1 \le i \le r$. The $r \cdot \ell$ encrypted messages $\{E(s_j, k_{i,j}) \ : \ 1 \le i \le r, 1 \le j \le \ell\}$ are broadcast. Finally, since each user knows exactly one key from each column, he can decrypt one of the r messages that contain s_j, for all j, and therefore recover the secret s.

Assume that $T = \{t_1, \ldots, t_p\} \subset U$ is a coalition of traitors. Then the arsenal of keys that they have at their disposal is

$$\mathbf{K} = \bigcup_{1 \le j \le \ell} \mathbf{K}_j, \qquad \text{where} \quad \mathbf{K}_j = \{k_{i,j} \ : \ i \in \mathbf{P}_j := \{P(t_1)_j, \ldots, P(t_p)_j\}\}.$$

They could manufacture a fully functional decoder if they install in it one key from $\mathbf{K}_j$, for all j. Namely, the set of all selections of ℓ keys that the pirate may make corresponds to the subset of codewords

$$\langle P(T) \rangle = \mathbf{P}_1 \times \cdots \times \mathbf{P}_\ell. \tag{1}$$

This is indeed the correct generation assumption in this context since if in the jth column the traitors have together, say, only two keys—$k_{i_1,j}$ and $k_{i_2,j}$—they have no choice other than selecting one of those two keys and placing it in the pirate decoder. They could not remove those keys altogether, for then the decoder would not be able to recover s_j; they could not deduce from those two keys the value of any other key from the jth column; and they could not combine those keys in any manner that would yield a different key that would be of any help in trying to decrypt the messages transmitting s_j.

Finally, the tracing algorithm in this context is by means of majority. When a pirate decoder is captured, the keys in it are read and compared with the personal keys of all users. The user that has the greatest number of matches is framed as a traitor. By choosing the parameters r and ℓ appropriately, taking into consideration the number of users n and an upper bound on the number of colluding traitors, p, it is possible to decrease the tolerance parameter ε as desired.

Example 2: Fingerprinting Digital Data. Consider a data provider that sells viewing rights of the latest digital features. Since digital material may be cloned many times without experiencing quality degradation, immoral users that paid for their copies might redistribute such copies for a bargain price in the black market. In order to deter such users and to be able to trace them in case that they do exercise such piracy, Boneh and Shaw [3] suggested a fingerprinting technique. As in our framework, the original data is personalized prior to distribution. The movie V is broken up to ℓ short segments, $V = V_1 || \cdots || V_\ell$, where $||$ denotes concatenation, and to each segment, V_j, r almost-identical variants are generated, $V_j \mapsto \{V_j^1, \dots, V_j^r\}$. As in the previous example, we let $\Sigma = \{1, \dots, r\}$ and then the personalization function is of the same sort, $P\colon U \to \Gamma \subset \Sigma^\ell$. If user $u \in U$ was assigned the codeword $P(u)$, then he will get the following version of the movie:

$$V(u) = V_1^{i(1)} || \dots || V_\ell^{i(\ell)}, \qquad \text{where} \quad i(j) = P(u)_j, \quad 1 \le j \le \ell.$$

Boneh and Shaw used a stronger generation assumption than Chor et al. They assumed that in segments where the embedded mark is detectable by the pirate, namely, a segment j where the pirate owns at least two different variants, the pirate could reproduce any of the r different variants of that segment, or render that mark unreadable. This may be formulated as follows:

$$\langle P(T) \rangle = \{w \in (\Sigma \cup \{?\})^\ell \ \text{ s.t. } w = P(t_1)|_R\}, \tag{2}$$

where $R = \{j : P(t_1)_j = \cdots = P(t_p)_j\}$. They called $\langle P(T) \rangle$ the feasible set of T.

Boneh and Shaw concentrated on the case $r = 2$ and designed watermarking schemes that are capable of tracing at least one of the colluding traitors that participated in producing the pirate version, with an error probability as small as desired.

Going back to the generation assumption (2), we note that in the binary case $r = 2$ it becomes quite similar to the previous assumption (1), the only difference being that (2) allows the complete removal of detectable marks. However, in the binary case, a detectable mark occurs in segments where the pirate has all $r = 2$ variants. In that case there is not much point in the pirate removing the mark (an operation that usually damages the quality of the produced copy); instead, he could pick any of the $r = 2$ variants that he has. Indeed, in accord with this observation, the Boneh–Shaw scheme sets unreadable marks arbitrarily to zero. That closes entirely the gap between their generation assumption (2) and the previous one (1).

We note that when $r > 2$ assumption (2) is quite far-reaching in the context of a digital video. For example, Cox et al. [6] have introduced methods to generate secure video watermarks. Those watermarks could not be easily removed, unless the pirate edits out the entire segment. In addition, if the pirate got at some segment k distinct variants,

$1 < k < r$, he could not reproduce from them any of the remaining $r - k$ variants for that segment. Hence, when such secure watermarks are used, the strong generation assumption (2) may be relaxed to (1).

As a concluding remark on the generation assumption (2), we note that this is the relevant assumption in the centuries-old context of protecting logarithm tables. When such tables were protected by means of introducing random errors in the least significant digits, a pirate could detect (some of) the digits that were used for that matter and then change their value as he pleased.

With this general framework, we are now ready to describe the dynamic setting, that also complies to the same framework. In a dynamic data distribution setting the time dimension is introduced. That is, as opposed to static settings where primary and secondary data distributions were single actions that may be viewed as "point events" in time, in the dynamic setting they are continuous long-lasting events. The legal users depend on the center for continuous delivery of data, be it content (such as TV broadcasts in Pay-TV systems or Web pages in the case of Internet service providers) or periodical session keys. The pirate retransmits this data to his dependent customers. Hence, the secondary distribution exists in parallel to the primary one, with some delay. In order to combat piracy scenarios in this setting, one may adopt similar techniques to those in the static setting. However, since the feedback from the secondary distribution network is available, it may be used in order to adapt codeword marking per user in subsequent time steps. This capability is what distinguishes dynamic from static traitor tracing.

Despite those differences, the above described framework, that was formulated with static settings in mind, is adequate also for the dynamic setting, mutatis mutandis:

1. In static settings the code length, ℓ, and the size of the alphabet, r, are determined beforehand. In dynamic settings ℓ stands for the number of time steps until the scheme closes on all traitors and disconnects them, or forces them to become dormant. As for r, it is being updated along the search according to the findings of the scheme.
2. Static schemes determine the entire codeword of each user prior to primary distribution. Dynamic schemes, however, determine the jth letter in the codeword of each user only after receiving the feedback from the secondary distribution in the previous $(j - 1)$th round.

We proceed with a formal definition of dynamic traitor tracing schemes. Let $r = r(j)$ denote the number of variants that are used in time step j, $j \geq 1$; namely, the marking alphabet in that time step is $\Sigma_j = \{1, \ldots, r(j)\}$. The scheme decomposes U into $r(j)$ disjoint subsets, $U = \bigcup_{1 \leq i \leq r(j)} S_{i,j}$, and assigns to all users in $S_{i,j}$ variant number i of data segment j. It then waits to see what variant is being transmitted by the pirate in the secondary distribution network. Based on that feedback, s_j, and possibly also the entire pirate transmission up to that stage, $(s_1, \ldots, s_j)$, the scheme:

- Decides whether any of the users may be accused at this stage as being a traitor and be disconnected. If so, that user is removed from U.
- Determines the number of variants to be used in the next step, $r(j + 1)$.
- Reassigns variants to users.

The first study of dynamic traitor tracing [7] offered two schemes. The first scheme uses $2p+1$ variants, where p stands for the actual number of active traitors, and converges in optimal time; it is described in Section 2 as it is the basis for the scheme that we present there. The second scheme uses $p+1$ variants, which is the minimal size for the marking alphabet that allows deterministic tracing in view of Theorem 1 of [7]; the convergence time of that scheme, however, depends exponentially on p, hence it can be efficiently implemented only for very small coalitions. Berkman et al. [2] offered a wide array of efficient dynamic traitor tracing schemes. They devised schemes with an alphabet of size $p+c+1$, for $1 \leq c \leq p$, with a running time of $O(p^2/c + p \log n)$. They also presented algorithms that used an alphabet of size $pc+1$ that converge in $O(p \log_c n)$ time steps.

All of those schemes use an alphabet of size $p+1$ at the least. When the pirate controls a large number of traitors, say $p \sim 100$, this duplication factor may be too expensive to pay. (In Section 4 we discuss piracy scenarios in Pay-TV systems and consider the implications of such a toll on the bandwidth in each of those scenarios.) Therefore, it is advisable to devise dynamic traitor tracing schemes that use a small fixed-size marking alphabet. This translates to a very important advantage in implementing traitor tracing: when embarking upon a deterministic traitor tracing, the tracer does not know in advance how much extra bandwidth is needed to support the overhead of multiple variants. The pirate could always surprise the tracer by activating yet another traitor. As mentioned earlier, for large, yet reasonable values of p, the tracer could meet the limits of his bandwidth capabilities. Schemes that use fixed-size marking alphabets release us from such worries.

This significant advantage is accompanied by two inherent disadvantages:

(a) As deterministic scheme must use an alphabet of size $p+1$ at least, such schemes would have to be probabilistic. Practically, this means that the center should use the findings of the scheme as a very strong evidence against the framed user, but it would have to conduct some further investigation in order to obtain physical evidence against that user.

(b) Such schemes would be slower than schemes that use richer alphabets.

The paper is organized as follows. In Section 2 we present a dynamic scheme with binary marks that is obtained by combining a deterministic dynamic scheme due to Fiat and Tassa [7] with the Boneh–Shaw scheme. We also derive upper bounds for its convergence time and error probability. In Section 2.4 we discuss the performance of that scheme and compare it with a static scheme that was presented in [3]. We show that the convergence time of our scheme is much better than that of the static one and that this performance-gap stems mainly from the dynamic nature of our scheme. In Section 3 we discuss other primitives that could serve as the inner or outer schemes, including a recent binary fingerprinting scheme due to Tardos [10] that improves the Boneh–Shaw scheme in an optimal manner. We also suggest a direction that should be explored in order to find more efficient dynamic binary schemes. Finally, in Section 4 we discuss implementation issues in the context of Pay-TV.

2. A Dynamic Scheme with Binary Marks

Here, we show how to combine the time-efficient dynamic traitor tracing scheme of Section 3.3 of [7] with the Boneh–Shaw scheme (BSS henceforth) in order to yield a probabilistic dynamic traitor tracing scheme that uses a binary alphabet. The first two schemes are described in Sections 2.1 and 2.2; the dynamic hybrid scheme is presented and analyzed in Section 2.3; finally, in Section 2.4, we compare the performance of our dynamic hybrid scheme with that of a static hybrid scheme that was presented in Section 5.1 of [3].

2.1. *A Deterministic Dynamic Traitor Tracing Scheme*

This scheme maintains a parameter t that denotes a proven lower bound on the number of active traitors; it is initialized to $t = 0$ and is updated according to the findings of the scheme. The set of subscribers, U, is partitioned into $2t + 1$ subsets,

$$U = \bigcup_{S \in P} S, \qquad \text{where} \quad P = \{L_1, R_1, \ldots, L_t, R_t, \ I\}, \tag{3}$$

and each of those sets receives a unique variant. An invariant of the algorithm is that the union $L_i \cup R_i$ contains at least one traitor for all $1 \le i \le t$, where I is the complementary subset of users that is not known to include a traitor. Hence, there are never more than $2p+1$ simultaneous variants, where p is the true number of traitors. Whenever a variant of one of the subsets in P is distributed by the pirate, that subset is split into two subsets in order to close on the traitors that it includes. If that subset is a singleton, the single user in it is framed and disconnected. As shown in Theorem 2 of [7], this scheme traces all p traitors and disconnects them within

$$L = p \cdot (\log n + 1) \tag{4}$$

time steps, which is shown to be optimal in [7] and [2].

2.2. *The Boneh–Shaw Scheme*

BSS assigns binary codewords to all n users so that, no matter what the size of the coalition of traitors, any pirate copy may be traced back to one of the users that participated in producing it with an error probability of no more than ε, for an arbitrary $\varepsilon > 0$. The code is defined as follows:

1. Set $d = d(n, \varepsilon) = 2n^2 \ln(2n/\varepsilon)$.
2. Set the length of the codewords to be $d \cdot (n-1)$, namely,
$$\ell = \ell(n, \varepsilon) = 2n^2(n-1)\ln(2n/\varepsilon). \tag{5}$$
3. Select a secret random permutation $\pi \in S_\ell$.
4. Define the n words $w^i = 0^{d(i-1)}1^{d(n-i)} \in \{0,1\}^\ell$, $1 \le i \le n$.
5. Assign to user u_i the codeword πw^i.

The tracing algorithm works as follows [3, Algorithm 1]: Let $\pi x \in \{0,1\}^\ell$ be the codeword found in a pirate copy, let $B_i = (x_{(i-1)d+1}, \ldots, x_{id})$ be the ith block of consecutive d bits in x and let k_i denote the number of ones in B_i, $1 \le i \le n-1$.

- If $k_1 > 0$ then user 1 is guilty (with certainty).
- If $k_{n-1} < d$ then user n is guilty (with certainty).
- For $2 \le i \le n-1$, if $k_{i-1} < k/2 - \sqrt{(k/2)\ln(2n/\varepsilon)}$, where $k = k_{i-1} + k_i$, then user i is guilty.

2.3. *The Hybrid Scheme*

We denote the dynamic traitor tracing scheme by $\mathcal{S}$ and the ε-secure BSS for n users by $\Gamma_\varepsilon = \Gamma_\varepsilon(n)$. We now combine those two into a probabilistic dynamic scheme, denoted $\mathcal{S} \circ \Gamma_\varepsilon$, that uses a binary watermarking alphabet.

The Scheme $\mathcal{S} \circ \Gamma_\varepsilon$.

1. Use the deterministic traitor tracing scheme $\mathcal{S}$ as an external basic scheme. The steps of $\mathcal{S}$ are referred to as *time segments*. Each *time segment* will be executed by applying several *time steps* as explained below.
2. If at the beginning of the jth time segment of $\mathcal{S}$ there are $2t_j + 1$ subsets, apply $\Gamma_\varepsilon(2t_j + 1)$. The application of that fingerprinting scheme as an inner procedure lasts $\ell = \ell(2t_j + 1, \varepsilon)$ *time steps*, where ℓ is given in (5).
3. At the end of the $\Gamma_\varepsilon(2t_j + 1)$ procedure, apply the tracing algorithm [3, Algorithm 1] in order to decide which of the $2t_j + 1$ subsets includes a traitor.
4. Proceed along the lines of $\mathcal{S}$.

Normally, the number of time segments in $\mathcal{S}$ until it traces and disconnects all traitors is given by (4). However, when in each segment we rely upon the probabilistic method Γ_ε to find the subset that includes a traitor, there is a chance of making wrong decisions that would split innocent subsets and add unnecessary time segments. Therefore, T—the actual number of time segments in $\mathcal{S} \circ \Gamma_\varepsilon$ until it traces and disconnects all true p traitors—might be larger than L, (4). In the next lemma we provide an almost certain upper bound on the convergence time of our probabilistic scheme.

Lemma 1. *Let $\lambda \geq 2$ be an arbitrary integer for which ε satisfies*

$$\varepsilon \leq \left(\frac{\lambda}{L_\lambda e}\right)^{\lambda/(\lambda-1)}, \qquad \textit{where} \quad L_\lambda = (p + \lambda) \cdot (\log n + 1). \tag{6}$$

Then

$$\mathrm{Prob}(T \leq L_\lambda) > (1 - \varepsilon), \tag{7}$$

where T is the number of time segments until $\mathcal{S} \circ \Gamma_\varepsilon$ traces and disconnects all p traitors.

Proof. Let x_j be an indicator random variable that equals one if BSS made a wrong accusation of a subset as being infected at time segment j. As we implement in that time segment the scheme $\Gamma_\varepsilon(2t_j + 1)$, we conclude that

$$\mathrm{Prob}(x_j = 1) \leq \varepsilon. \tag{8}$$

Note that even when BSS wrongly accuses a subset, that subset might be infected with traitors that were dormant during that time segment. In that case we set $x_j = 0$, in order to let $x_j = 1$ denote only "bad" time segments.

Let us assess the damage that is incurred by bad time segments j where $x_j = 1$. Assume that BSS wrongly accused the subset L_i, $1 \leq i \leq t$, as being infected. Then, consequently, the adjacent subset R_i, that might be infected, will be absorbed in I. That way we lose the information that we already gained on the position of the traitors that were trapped in R_i. Obviously, the smaller are the subsets L_i and R_i, the larger the penalty that we have to pay in convergence time. If such an error occurs in the worst case when $|L_i| = 1$, the penalty in convergence time is maximal and, in addition, we also accuse and disconnect an innocent user. It should be noted that if the wrongly accused subset is I, the penalty is minimal since we do not lose any information and no wrong accusations are made.

Let T denote the number of time segments until all traitors (and, possibly, also some other innocent users) are traced and disconnected. Let $\lambda = \sum_{j=1}^{T} x_j$ denote the overall number of BSS errors. In the worst case, where all λ errors happened with singleton subsets, we have $T = L_\lambda$ since we may view the λ innocent users as λ dormant traitors. Hence, in general, $T \le L_\lambda$. Therefore, for a given λ, we conclude that $T > L_\lambda$ implies that

$$\sum_{j=1}^{L_\lambda} x_j > \lambda. \tag{9}$$

Hence,

$$\mathrm{Prob}(T > L_\lambda) \le \mathrm{Prob}\left(\sum_{j=1}^{L_\lambda} x_j > \lambda\right). \tag{10}$$

Using the Chernoff bound we get from (8) and (10) that

$$\mathrm{Prob}(T > L_\lambda) \le \left(\frac{e^{\beta-1}}{\beta^\beta}\right)^{\varepsilon L_\lambda}, \qquad \text{where} \quad \beta = \frac{\lambda}{\varepsilon L_\lambda}. \tag{11}$$

However, in view of (6),

$$\left(\frac{e^{\beta-1}}{\beta^\beta}\right)^{\varepsilon L_\lambda} < \left(\frac{e}{\beta}\right)^{\lambda} = \left(\frac{e\varepsilon L_\lambda}{\lambda}\right)^{\lambda} \le \varepsilon. \tag{12}$$

The desired estimate (7) now follows from (11) and (12). □

After estimating the number of time segments in our scheme, i.e., the number of rounds in the outer scheme $\mathcal{S}$, we turn to estimating the number of time steps until the scheme traces all traitors and the probability of making false accusations.

Theorem 1. *Let $\lambda \ge 2$ be an arbitrary integer and let $\varepsilon > 0$ be a parameter satisfying* (6). *Then the scheme $\mathcal{S} \circ \Gamma_\varepsilon$ will detect and disconnect all p traitors within M time steps where*

$$\mathrm{Prob}(M = O(p^4 \ln(p/\varepsilon) \log n)) > 1 - \varepsilon. \tag{13}$$

Moreover, the probability of incriminating and disconnecting an innocent user (*in addition to the p true traitors*) *is at most $L\varepsilon$, where $L = p \cdot (\log n + 1)$.*

Proof. Let t_j be the lower bound on the number of traitors in time segment j, as inferred by the outer scheme $\mathcal{S}$. In view of Lemma 1,

$$\text{Prob}\left(\max_j t_j \le p + \lambda\right) > 1 - \varepsilon, \tag{14}$$

where p is the number of true traitors. Let M_j denote the number of time steps in time segment j. Then, by (14) and (5), the following bound holds for all j in probability of at least $(1 - \varepsilon)$:

$$M_j \le \ell(2t_j + 1, \varepsilon) < 2r^3 \ln(2r/\varepsilon), \qquad \text{where} \quad r = 2(p + \lambda) + 1. \tag{15}$$

Consequently, the number of time steps until the scheme detects and disconnects all traitors, $M = \sum_{j=1}^{L_\lambda} M_j$, is bounded as follows:

$$\text{Prob}(M \le L_\lambda \cdot 2r^3 \ln(2r/\varepsilon)) > 1 - \varepsilon. \tag{16}$$

In view of (6), (15) and (16) we conclude that

$$M < 16(p + \lambda + \tfrac{1}{2})^3 \cdot \ln\left(\frac{4(p + \lambda + \frac{1}{2})}{\varepsilon}\right) \cdot (p + \lambda) \cdot (\log n + 1), \tag{17}$$

in probability of at least $1 - \varepsilon$. Rearranging the upper bound in (17), we get that

$$M \le 16\gamma_1 \cdot p^4 \cdot \ln\left(\frac{4(p + \lambda + \frac{1}{2})}{\varepsilon}\right) \cdot \left(1 + \frac{\lambda}{p}\right) \cdot (\log n + 1)\,, \tag{18}$$

where

$$\gamma_1 = \left(1 + \frac{\lambda + \frac{1}{2}}{p}\right)^3. \tag{19}$$

This shows that the scheme will detect and disconnect all p true traitors within M time steps where $M = O(p^4 \ln(p/\varepsilon) \log n)$, in probability of at least $1 - \varepsilon$.

Regarding the probability of falsely incriminating and disconnecting an innocent user, we recall that assumption (6) guarantees that the number of wrong decisions along the search is bounded by λ with probability at least $1 - \varepsilon$. We note that wrongly accusing a subset as being infected when the size of that subset is greater than one, affects only the number of time segments until complete convergence on all traitors is achieved. However, wrongly accusing a singleton subset as being infected causes an additional damage: an innocent user would be incriminated and disconnected. In order to estimate the probability of such a misfortune, we introduce the following event notations:

- A denotes the event of completing the entire tracing scheme without disconnecting any innocent user. Note that the event A does allow for BSS errors during the search; however, it does not allow such errors to occur with singleton sets of innocent users.
- E_i denotes the event of having exactly i BSS errors until the completion of the search.

With these notations, we see that

$$\mathrm{Prob}(A) = \sum_{i=0}^{\infty} \mathrm{Prob}(A \mid E_i) \cdot \mathrm{Prob}(E_i) \geq \mathrm{Prob}(A \mid E_0) \cdot \mathrm{Prob}(E_0). \tag{20}$$

Since $\mathrm{Prob}(A \mid E_0) = 1$ and $\mathrm{Prob}(E_0) \geq (1-\varepsilon)^L$, we conclude that

$$\mathrm{Prob}(A) \geq 1 - L\varepsilon. \tag{21}$$

This completes the proof. □

The lower bound in (21) may seem poor since it relies upon the first term only in the infinite sum in (20). However, it cannot be improved significantly, as illustrated by deriving an approximate upper bound for $\mathrm{Prob}(A)$: Letting $\hat{E}_\lambda = \bigcup_{i=0}^{\lambda} E_i$ denote the event of having no more than λ errors until the completion of the search, and letting $\hat{E}_\lambda^{\mathrm{c}}$ denote its complement, we get that

$$\mathrm{Prob}(A) = \mathrm{Prob}(A \mid \hat{E}_\lambda) \cdot \mathrm{Prob}(\hat{E}_\lambda) + \mathrm{Prob}(A \mid \hat{E}_\lambda^{\mathrm{c}}) \cdot \mathrm{Prob}(\hat{E}_\lambda^{\mathrm{c}}). \tag{22}$$

In view of Lemma 1, $\mathrm{Prob}(\hat{E}_\lambda) > (1-\varepsilon)$. Using this lower bound as an approximation for the value of $\mathrm{Prob}(\hat{E}_\lambda)$, we get that

$$\begin{aligned}\mathrm{Prob}(A) &\approx \mathrm{Prob}(A \mid \hat{E}_\lambda) \cdot (1-\varepsilon) + \mathrm{Prob}(A \mid \hat{E}_\lambda^{\mathrm{c}}) \cdot \varepsilon \\ &\leq \mathrm{Prob}(A \mid \hat{E}_\lambda) \cdot (1-\varepsilon) + \varepsilon. \end{aligned} \tag{23}$$

Let u denote the first innocent user that appears as a singleton set in the search (note that such a singleton set must appear at some point; in case more than one set of that sort appears, for the first time, simultaneously, we let u denote one of them). Then

$$\mathrm{Prob}(A \mid \hat{E}_\lambda) \leq \mathrm{Prob}(A_u \mid \hat{E}_\lambda), \tag{24}$$

where A_u is the event of never incriminating u. Next, let t_0 denote the time segment in which u popped up as a singleton set, let t_∞ denote the time segment when the search has ended and, for every $t_0 \leq i \leq t_\infty$, let b_i be the number of innocent leaves in the search tree at time segment i (i.e., how many subsets in the partition P, (3), do not include a traitor). With these notations we conclude that

$$\mathrm{Prob}(A_u \mid \hat{E}_\lambda) \approx \prod_{i=t_0}^{t_\infty} \left(1 - \frac{\varepsilon}{b_i}\right). \tag{25}$$

The number of innocent leaves that face the danger of being wrongly incriminated may be bounded as follows:

$$b_i \leq p + 1 + \lambda, \qquad t_0 \leq i \leq t_\infty, \tag{26}$$

because in the error-free scheme there are always no more than $p+1$ innocent leaves, while any wrong decision (the number of which is assumed to be no more than λ) may increase the number of innocent leaves by one at the most. Hence, in view of (24)–(26),

$$\mathrm{Prob}(A \mid \hat{E}_\lambda) \leq \left(1 - \frac{\varepsilon}{p+1+\lambda}\right)^{t_\infty - t_0}. \tag{27}$$

Next, as $t_\infty \geq L$, we get that

$$\mathrm{Prob}(A \mid \hat{E}_\lambda) \leq \left(1 - \frac{\varepsilon}{p+1+\lambda}\right)^{L-t_0}. \tag{28}$$

Regarding t_0, the event of u popping up as a singleton set may happen no sooner than after $\log n$ time segments. Hence, $t_0 \geq \log n$. It is clear that as long as there are no singleton sets in the search tree, there is no danger of harming innocent users. Hence, the worst scenario is that in which the search tree develops in the most unbalanced manner, in order to introduce singleton sets as early as possible. This is also apparent from (28). Hence, it is in the interest of the pirate to activate only one of its traitors at the first $\log n$ stages, in order to decrease t_0 to a minimum. Therefore, assuming such a pirate strategy, we get that

$$\mathrm{Prob}(A \mid \hat{E}_\lambda) \leq \left(1 - \frac{\varepsilon}{p+1+\lambda}\right)^{L-\log n}. \tag{29}$$

Hence, by (23) and (29),

$$\begin{aligned}\mathrm{Prob}(A) &\leq \left(1 - \frac{\varepsilon}{p+1+\lambda}\right)^{L-\log n} \cdot (1-\varepsilon) + \varepsilon \\ &= 1 - (L - \log n) \cdot \frac{\varepsilon}{p+1+\lambda} + O(\varepsilon^2).\end{aligned} \tag{30}$$

Therefore, we arrive at the following approximate lower bound for the probability of the complement event, assuming a malicious pirate policy:

$$\mathrm{Prob}(A^{\mathrm{c}}) \geq (L - \log n) \cdot \frac{\varepsilon}{p+1+\lambda} + O(\varepsilon^2). \tag{31}$$

Comparing this to the upper bound $\mathrm{Prob}(A^{\mathrm{c}}) \leq L\varepsilon$, (21), we see that they differ by an approximate factor of $1/p$.

For the sake of performance comparison, it is convenient to rescale ε so that the probability of falsely accusing an innocent user is bounded by exactly ε. The required rescaling factor is $L = p \cdot (\log n + 1)$, which involves the unkown number of traitors p. Hence, assuming that c is a safe upper bound for the number of traitors, we rescale ε using the larger factor $\hat{L}_0$ where

$$\hat{L}_\lambda = (c + \lambda) \cdot (\log n + 1). \tag{32}$$

With this, we may state the following consequence of Theorem 1.

Theorem 2. *Assume that c is an upper bound for the number of traitors and let ε be the required tolerance parameter of the scheme. Let $\lambda \geq 2$ be an arbitrary integer for which*

$$\varepsilon < \hat{L}_0 \cdot \left(\frac{\lambda}{\hat{L}_\lambda e}\right)^{\lambda/(\lambda-1)}, \tag{33}$$

where $\hat{L}_\lambda$ is given in (32). *Then the scheme $\mathcal{S} \circ \Gamma_{\varepsilon/\hat{L}_0}$ will detect and disconnect all p traitors within M time steps where* $\text{Prob}(M \le M_d) \ge 1 - \varepsilon/\hat{L}_0$,

$$M_d = 16\gamma_1 \cdot p^4 \cdot \ln\left(\frac{4\hat{L}_0(p + \lambda + \frac{1}{2})}{\varepsilon}\right) \cdot \left(\log n + 1 + \frac{\lambda}{p}\right), \tag{34}$$

and γ_1 is given in (19). *Moreover, the probability of incriminating and disconnecting an innocent user* (*in addition to the p true traitors*) *is at most ε.*

Remark. The parameter λ has a very minor significance. In fact, if $\varepsilon < 0.1$, assumption (33) holds for all $\lambda \ge 2$.

2.4. *Performance Comparison*

The idea of combining a binary code with a richer alphabet code in order to combine the bandwidth efficiency of the former with the time efficiency of the latter already appeared in Section 5.1 of [3]. The static scheme presented there was obtained by the composition of BSS with the simplest traitor tracing scheme of Section 5.1 of [5]. This latter scheme, to which we refer henceforth as the CFN scheme, assumes an upper bound on the number of traitors, c, and then distributes codewords to users from Σ^ℓ where $|\Sigma| = 2c$ and $\ell = \gamma_2 c \log(n/\varepsilon)$, with

$$\gamma_2 = \frac{2\ln 2}{2\ln 2 - 1}. \tag{35}$$

(The CFN scheme, as presented in Section 5.1 of [5], uses an alphabet of size $4c$; however, it may use any alphabet of size ηc where η is a constant greater than one. The example in Section 5.1 of [3] employs CFN with $\eta = 2$, as we do here.) With this choice of an outer scheme, the inner scheme is $\Gamma_{\varepsilon/2\ell}(2c)$, where ℓ is given above. Hence, the length of its codewords is $d \cdot (2c - 1)$ where $d = 8c^2 \ln(8c\ell/\varepsilon)$. Multiplying the length of the outer codewords over Σ by the length of the inner binary encoding, we get the overall length of codewords generated by that scheme:

$$M_s = \ell \cdot d \cdot (2c - 1) \le 16\gamma_2 c^4 \ln\left(\frac{8\gamma_2 c^2 \log(n/\varepsilon)}{\varepsilon}\right) \log\left(\frac{n}{\varepsilon}\right). \tag{36}$$

It may seem at first that M_s, the length of the static scheme, (36)+(35), is of the same order of magnitude as that of the dynamic scheme, M_d, (34)+(19). However, this is not the case, as we clarify in the comparative discussion below.

- The upper bound c

Both our scheme and the above-described static scheme need an a priori bound on the number of traitors, c. Our dynamic scheme needs it only for the sake of setting the parameter $\varepsilon/\hat{L}_0$ to be used in the inner BSS (the outer scheme does not need to know c since it learns the actual number of traitors, p, along the run). The static scheme, however, needs to know c for the sake of the outer CFN scheme.

If our scheme uses a too low c, i.e., $c < p$, the error probability will become greater than the desired one—ε (say, if $c = p/2$, the actual probability error would be 2ε). If, however, the CFN scheme uses any value of $c \leq p$, it completely collapses and will incriminate innocent users.

- The dependence of the run time on c

As a consequence of the above, c must be set to a safe a priori bound (where the CFN-based static scheme must be significantly more “cautious” than the dynamic scheme). Such an a priori bound could be several times larger than p, the actual number of active traitors. The resulting toll on the number of time steps in the static scheme could be devastating due to the fourth power (c^4 in (36) versus p^4 in (34)). The run-time of the dynamic scheme, however, depends only on $\ln c$.

- Complete tracing versus partial tracing

The dynamic scheme is guaranteed to trace and disconnect **all** p traitors within M_d time steps, with probability at least $1 - \varepsilon/\hat{L}_0$. The static scheme, however, is guaranteed to trace and disconnect **at least one** of the traitors, with probability at least $1 - \varepsilon$. Hence, if we utilize the static scheme in the dynamic setting, we would need to apply it p times until piracy stops! Hence, the actual run time of that scheme when used in the dynamic setting is $p \cdot M_s$. Since p may easily be in the hundreds, that is yet another crucial factor in favor of the dynamic scheme.

Example. Consider a system with $n = 10^5$ users where a coalition of size $p = 10$ is exercising piracy. Assume that we wish to trace all traitors with an error probability of no more than $\varepsilon = 10^{-3}$ and that $c = 50$ is the assumed bound for the size of the coalition.

The static scheme. The length of codewords in that scheme turns out to be $M_s \approx 2.04 \cdot 10^{11}$. The actual run-time until we trace all traitors and stop piracy would be therefore $p \cdot M_s \approx 2.04 \cdot 10^{12}$.

The dynamic scheme. The upper bound $c = 50$ gives us $\hat{L}_0 \approx 880$. With $\lambda = 2$ we get that $M_d \approx 10^8$. This run-time is four orders of magnitude lower than that of the static scheme.

3. Alternative Primitives for the Inner and Outer Schemes

The above example illustrates the difficulty of traitor tracing in practice. While the deterministic schemes of [7] and [2] may collapse due to devastating bandwidth requirements, the above scheme is bandwidth-efficient, but it requires an intolerable amount of time to converge. This inefficiency is due to the inefficiency of the inner BSS (the outer dynamic scheme runs in an optimal time). The problematic factor in the run-time of BSS is the third power that appears in the codeword length of BSS, see (5) and (15). Boneh and Shaw established a lower bound on the length of codewords in a fingerprinting scheme over a binary alphabet. They showed that given a binary fingerprinting scheme that is ε-secure against coalitions of size c at the most, the length of its codewords is bounded from below by $\frac{1}{2}(c-3)\log(1/\varepsilon c)$ [3, Theorem 6.1]. This lower bound was improved by Peikert et al. to $\Omega(c^2 \log(1/\varepsilon c))$ [9]. After the acceptance of this manuscript for publication, Tardos presented a paper [10] where he closes the gap by offering a fingerprinting code of size n that is ε-secure against coalitions of size c, where the length of the code is $O(c^2 \log(n/\varepsilon))$. That code is optimal to within a constant factor for small error probabilities. Setting $c = n$ we get a code that is ε-secure against *any* coalition with length $O(n^2 \log(n/\varepsilon))$. Comparing this with (5) we infer that utilizing the Tardos code instead of BSS as the inner scheme, we can shave off a factor of $O(p)$ in the convergence time of the hybrid scheme, (34).

Next, we consider other choices for the outer scheme. Berkman et al. presented in [2] an array of time-efficient dynamic traitor tracing schemes. They showed that using $p + c$ variants, for any $1 \leq c \leq p + 1$, it is possible to locate all p traitors within $O(p^2/c + p \log n)$ rounds. The largest value $c = p + 1$ corresponds to the scheme presented in [7] that we used in Section 2. The scheme that corresponds to the smallest value $c = 1$ is optimal in the sense that it uses the minimal number of variants that allows deterministic detection: $p + 1$. The price that we need to pay for decreasing the number of variants from $2p + 1$ to $p + 1$ is twofold: (a) the run time increases from $p \cdot (\log n + 1)$ to $O(p \log n + p^2)$; and (b) the algorithm becomes significantly more intricate. The dominant factor in M_d, (34), is $16p^4$. What gave rise to that factor is the length of each time segment, namely, the length of codewords in the inner scheme. That length depended on $2r^3$ (see (5) and (15)) and since in the outer scheme $r \approx 2p$, we get the factor $16p^3$ (the additional p that raises the power from 3 to 4 comes from (4)). Hence, if we replace the outer scheme with the scheme that uses only $r = p + 1$ variants we are expected to save a constant factor of $2^3 = 8$. However, the constants hidden in the run-time of that scheme, $O(p \log n + p^2)$, render that choice unattractive. Therefore, in the above-mentioned range of deterministic schemes, $1 \leq c \leq p + 1$, the upper extreme one yields the most efficient as well as simplest binary scheme.

Another range of schemes studied in [2] corresponds to schemes that use $r = pc + 1$ variants, where $c \geq 2$. Their run-time is $O(p \log_c n)$. Here it is quite obvious that the choice $c = 2$ yields the most efficient binary scheme. In view of the above, it seems that our choice of the $(2p + 1)$-variant deterministic scheme is best.

We conclude this section with the following observation: The recipe that we presented here in order to obtain dynamic schemes over binary alphabets may be too restricted. That recipe uses a static scheme during any time segment; namely, the feedback that is collected during a time segment is analyzed only at the end of that time segment. Another direction that should be explored is that of *fully dynamic* schemes over small alphabets, where the feedback in each *time step* is taken into account immediately.

4. Traitor Tracing in Pay-TV Systems

The most notable example of a dynamic data distribution system is that of Pay-TV. In order to assure that users pay for content in such systems, *Conditional Access* techniques are used. Conditional Access techniques work as follows:

- *Level* 1. The content is encrypted using encryption keys that are called *Control Words*.
- *Level* 2. The control words are delivered to all users in encrypted messages, called *Entitlement Control Messages*, or *ECMs*, where the encryption key is a common periodical key.

- *Level* 3. The periodical keys are refreshed on a regular basis and their value is delivered to all users in encrypted messages. In the simplest setting, those messages would be personally addressed and encrypted with the personal key that is installed in the addressee decoder. However, due to bandwidth considerations, the value of the periodical key may have to be delivered to each and every user in more elaborated techniques.

Piracy occurs when a legal user (that may have registered as a customer several times under different identities) performs one of the following actions:

- *Scenario* 1. Redistributes the decrypted content to illegal customers.
- *Scenario* 2. Redistributes the control words that encrypt the content.
- *Scenario* 3. Redistributes the periodical keys that encrypt the control words.

All those scenarios comply with the traitor tracing framework, Section 1. The first scenario is the least probable: it seems unlikely that the pirate would rebroadcast the MPEG-II material over the Internet, due to bandwidth limitations. Rebroadcasting the content via cable or terrestrial networks, on the other hand, might expose the pirate. In any case, if piracy occurs on that level, the content would have to be marked (say, by techniques similar to those presented in [6]) in order to enable the activation of any traitor tracing scheme. As the transmission of content requires huge amounts of bandwidth, only a small fraction of the content could be marked (say, 1 channel only, and only 5 seconds in each minute) and the duplication factor should be limited to $O(1)$.

The second scenario is also quite hard for the pirate to undertake. Control words are refreshed every few seconds and the ECMs that deliver their value must be repeated very quickly in order to allow a reasonable switch delay in the decoder. Those requirements may be too large for the bandwidth limitations of the pirate. In the case where piracy occurs on this level, the tracer would have to duplicate the control words of one of the channels and, consequently, the entire encrypted transmission of the selected channel. Such an overhead might be intolerable, unless bandwidth efficient schemes, such as the ones presented here, are employed.

The third scheme is the most probable one. Here, the pirate has to retransmit only one periodical key in a very low rate. If piracy occurs on that level, the tracer would have to duplicate the periodical keys and, consequently, also the ECM stream. In this context, the deterministic schemes are applicable despite their bandwidth overhead. However, as the center might need to reallocate bandwidth from content channels to control channels transmitting ECMs, this translates to immediate financial loses as well as termination of services. Hence, the center might prefer to activate low-bandwidth schemes, despite their disadvantages in comparison with deterministic schemes, since they would not necessitate the termination of existing services.

Acknowledgments

I thank Omer Berkman for reading the manuscript and making insightful comments.

References

[1] N. Alon and J. Spencer, *The Probabilistic Method*, Wiley, New York, 1992.

[2] O. Berkman, M. Parnas and J. Sgall, Efficient Dynamic Traitor Tracing, *SIAM Journal on Computing*, vol. 30, no. 6 (2001), pp. 1802–1828. See also *Proc*. 11*th Annual ACM–SIAM Symposium on Discrete Algorithms* (*SODA*, 2000), pp. 586–595.

[3] 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, LNCS 963, Springer-Verlag, Berlin, 1995, pp. 452–465.

[4] B. Chor, A. Fiat, and M. Naor, Tracing Traitors, *Proc*. *Crypto* 94, LNCS 839, Springer-Verlag, Berlin, 1994, pp. 257–270. For a full version see [5].

[5] B. Chor, A. Fiat, M. Naor and B. Pinkas, Tracing Traitors, *IEEE Transactions on Information Theory*, vol. 46, no. 3 (2000), pp. 893–910.

[6] I.J. Cox, J. Kilian, T. Leighton and T. Shamoon, A Secure, Robust Watermark for Multimedia, in *Information Hiding*, LNCS 174, Springer-Verlag, Berlin, 1996, pp. 185–226.

[7] A. Fiat and T. Tassa, Dynamic Traitor Tracing, *Journal of Cryptology*, vol. 14, no. 3 (2001), pp. 211–223. See also *Proc*. *Crypto* 99, LNCS 1666, Springer-Verlag, Berlin, 1999, pp. 537–554.

[8] P. Kocher, Timing Attacks on Implementations of Diffie–Hellman, RSA, DSS, and Other Systems, *Proc*. *Crypto* 96, LNCS 1109, Springer-Verlag, Berlin, 1996, pp. 104–113.

[9] C. Peikert, A. Shelat and A. Smith, Lower Bounds for Collusion-Secure Fingerprinting, *Proc*. 40*th Symposium on Discrete Algorithms* (*SODA* '03), pp. 472–479.

[10] G. Tardos, Optimal Probabilistic Fingerprint Codes, *Proc*. 35*th ACM Symposium on Theory of Computing* (*STOC* '03), pp. 116–125.