Estimating the minimal length of Tardos code

Teddy Furon[1]*, Luis Pérez-Freire[2], Arnaud Guyader[3,4], and Frédéric Cérou[3]

[1] Thomson Security Lab, Cesson-Sévigné, France
[2] GRADIANT - Galician R&D Center in Advanced Telecommunications, Vigo, Spain
[3] INRIA, Centre Rennes - Bretagne Atlantique, Rennes, France
[4] Université Rennes 2, Rennes, France

Abstract. This paper estimates the minimal length of a binary probabilistic traitor tracing code. We consider the code construction proposed by G. Tardos in 2003, with the symmetric accusation function as improved by B. Skoric *et al.* The length estimation is based on two pillars. First, we consider the Worst Case Attack that a group of c colluders can lead. This attack minimizes the mutual information between the code sequence of a colluder and the pirated sequence. Second, an algorithm pertaining to the field of rare event analysis is presented in order to estimate the probabilities of error: the probability that an innocent user is framed, and the probabilities that all colluders are missed. Therefore, for a given collusion size, we are able to estimate the minimal length of the code satisfying some error probabilities constraints. This estimation is far lower than the known lower bounds.

Traitor tracing, Fingerprinting, Tardos code, Rare event

1 Introduction

This article deals with traitor tracing which is also known as active fingerprinting, content serialization, user forensics or transactional watermarking. The typical application is as follows: A video on demand server distributes personal copies of the same content to n buyers. Some are dishonest users whose goal is to illegally redistribute a pirate copy. The rights holder is interested in identifying these dishonest users. For this purpose, a unique user identifier consisting on a sequence of m symbols is embedded in each video content thanks to a watermarking technique, thus producing n different (although perceptually similar) copies. This allows tracing back which user has illegally redistributed his copy. However, there might be a collusion of c dishonest users, $c > 1$. This collusion mixes their copies in order to forge a pirated content which contains none of the identifiers but a mixture of them.

The traitor tracing code invented by Gabor Tardos in 2003 [11] becomes more and more popular. This code is a probabilistic weak fingerprinting code, where the probability of accusing an innocent is not null. The decoding of this

* This work is supported in part by the French national programme "Securité ET INformatique" under project NEBBIANO, ANR-06-SETIN-009.

code is focused, in the sense that it states whether or not a given user is guilty. Its performances are usually evaluated in terms of the probability ϵ_1 of accusing an innocent and the probability of missing all colluders ϵ_2. Most of the articles dealing with the Tardos code aim at finding a tight lower bound of the length of the code. In his seminal work G. Tardos shows that, in order to guarantee that the probability of accusing an innocent is below ϵ_1, the inequality $m > Kc^2 \log n/\epsilon_1$ with $K = 100$ must be satisfied. Many researchers found the constant $K = 100$ not accurate. Better known approximations are, for instance, $K = 4\pi^2$ [10], $K = 38$ [1]. Other works propose more practical implementations of the Tardos code [8]. The reader will find a pedagogical review of this code in [5].

The goal of this paper is to propose yet another evaluation of the Tardos constant. Whereas the previous articles proposed values based on theoretical bounds, our approach is radically different because it is purely experimental. For a given length of code m, we estimate the probability of accusing an innocent and the probability of missing all colluders. This task is not easy because the probabilities to be estimated are very small. Indeed, a classical Monte-Carlo algorithm would take too long time. The algorithm we propose in Sec. 4 is the result of a collaboration between statistician experts in rare event analysis and watermarkers. It is very generic and much more efficient than a classical Monte Carlo estimator. It estimates the probability P that a set of data $\mathbf{x} \in \mathcal{X}$ with a known pdf $p_{\mathbf{X}}$ has a score $s(\mathbf{x})$, with $s(.) : \mathcal{X} \rightarrow \mathbb{R}$ a deterministic score function, bigger than a given threshold τ: $P = \Pr[s(\mathbf{x}) > \tau]$.

This experimental approach has also the advantage to be closer to what really matters in practice. All the aforementioned theoretical bounds are based on the mean of the accusation scores of the colluders. This is needed for mathematical tractability but it provides an upper bound of the probability of false negative in the 'Detect-one' case: if the mean of the accusation scores is below the threshold, then at least one colluder is not accused, hence a false negative. In our experimental setup, we estimate the probability that the maximum (resp. minimum) of the colluders scores is below the threshold. This is the exact definition of a false negative event for the 'Detect-one' strategy (resp. 'Detect-all' strategy) [7].

The second main idea of this paper is the Worst Case Attack. Sec. 2 presents the model supporting a collusion strategy which is compliant with the well known *marking assumption* [2]. Among these collusion strategies, it appears that some of them have a deeper impact on the accusation performances than others. This is quite surprising because the Tardos decoding was previously believed to be invariant against the collusion strategy [5]. The Worst Case Attack (WCA) is thus defined as the collusion strategy minimizing the accusation performances. In order to evaluate these performances in the broadest sense, Sec. 3 relies on the concept of achievable rate of a traitor tracing code introduced in [7] as the criterion to be minimized. The experimental assessment is then based on the WCA in order to estimate the true minimal Tardos constant K.

2 The Setup

2.1 Code generation

We briefly remind how the Tardos code is designed. The binary code $\mathbf{X}$ is composed of n sequences of m bits. The sequence $\mathbf{X}_j = (X(j,1), \cdots, X(j,m))$ identifying user j is composed of m independent binary symbols, with $\Pr_{X(j,i)}[x(j,i) = 1] = p_i$, $\forall i \in [m]$, with $[m]$ denoting $\{1, \ldots, m\}$. $\{P_i\}_{i\in[m]}$ are independent and identically distributed auxiliary random variables in the range $[0,1]$: $P_i \sim f(p)$. Tardos proposed the following pdf, $f(p) = (\pi\sqrt{p(1-p)})^{-1}$, which is symmetric around $1/2$: $f(p) = f(1-p)$. It means that symbols '1' and '0' play a similar role with probability p or $1-p$. The actual occurrences $\{P_i\}_{i\in[m]}$ of these random variables are drawn once for all at the initialization of the code, and they constitute its secret key.

2.2 Collusion process

Denote the subset of colluder indices by $\mathcal{C} = \{j_1, \cdots, j_c\}$, and $\mathbf{X}_{\mathcal{C}} = \{\mathbf{X}_{j_1}, \ldots, \mathbf{X}_{j_c}\}$ the restriction of the traitor tracing code to this subset. The collusion attack is the process of taking sequences in $\mathbf{X}_{\mathcal{C}}$ as inputs and yielding the pirated sequence $\mathbf{Y}$ as an output.

Fingerprinting codes have been first studied by the cryptographic community and a key-concept is the *marking assumption* introduced by Boneh and Shaw [2]. It states that, in its narrow-sense version, whatever the strategy of the collusion $\mathcal{C}$, we have $Y(i) \in \{X(j_1,i), \cdots, X(j_c,i)\}$. In words, colluders forge the pirated copy by assembling chunks from their personal copies. It implies that if, at index i, the colluders' symbols are identical, then this symbol value is decoded at the i-th chunk of the pirated copy.

This is what watermarkers have understood from the pioneering cryptographic work. However, this has led to misconceptions. Another important thing is the way cryptographers have modelized a host content: it is a binary string where some symbols can be changed without spoiling the regular use of the content. These locations are used to insert the code sequence symbols. Cryptographers assume that colluders disclose symbols from their identifying sequences comparing their personal copies symbol by symbol. The colluders cannot spot a hidden symbol if it is identical on all copies, hence the marking assumption.

In a multimedia application, the content is divided into chunks. A chunk can be a few second clip of audio or video. Symbol $X(j,i)$ is hidden in the i-th chunk of the content with a watermarking technique. This gives the i-th chunk sent to the j-th user. In this paper, we mostly address collusion processing where the pirated copy is forged by picking chunks from the colluders' personal copies. The marking assumption in the studied problem still holds but for another reason: as the colluders ignore the watermarking secret key, they cannot create chunks of content watermarked with a symbol they do not have. However, contrary to the original cryptographic model, this also implies that the colluders might not know which symbol is embedded in a chunk.

At the end of the paper, we also consider a content post-processing: After mixing their copies, the colluders apply a coarse compression for instance. We assume that this yields decoding errors at the watermarking layer. However, we suppose the colluders do not control or know where the errors appear, and at which rate. Therefore, the collusion consists in two independent processing: mixing of copies followed by a degradation of the watermarking layer.

2.3 Mathematical model

Our mathematical model of the collusion is essentially based on four main assumptions. The first assumption is the *memoryless* nature of the collusion attack. Since the symbols of the code are independent, it seems relevant that the pirated sequence $\mathbf{Y}$ also shares this property. Therefore, the value of $Y(i)$ only depends on $\{X(j_1, i), \cdots, X(j_c, i)\}$.

The second assumption is the *stationarity* of the collusion process. We assume that the collusion strategy is independent of the index i in the sequence. Therefore, we can describe it for any index i, and we will drop indexing for sake of clarity in the sequel.

The third assumption is the *exchangeable* nature of the collusion: the colluders select the value of the symbol Y depending on the values of their symbols, but not on their order. Therefore, the input of the collusion process is indeed the type of their symbols (*i.e.* the empirical probability mass function). In the binary case, this type is fully defined by the following sufficient statistic: the number $\Sigma(i)$ of symbols '1': $\Sigma(i) = \sum_{k=1}^{c} X(j_k, i)$.

The fourth assumption is that the collusion process may be deterministic (for instance, majority vote, minority vote), or *random* (for instance, the symbol pasted in the pirated sequence is decided upon a coin flip).

These four assumptions yield that the collusion attack is fully described by the following parameter: $\boldsymbol{\theta} = \{\theta_0, \ldots, \theta_c\}$, with $\theta_\sigma = \mathrm{Pr}_Y[1|\Sigma = \sigma]$. There is thus an infinity of collusion attacks, but we can already state that they all share the following property: The marking assumption enforces that $\theta_0 = 0$ and $\theta_c = 1$. A collusion attack is thus defined by $c - 1$ real values belonging to the hypercube $[0, 1]^{c-1}$.

At the end of the paper, we consider a content processing on top of the collusion. The model is a Binary Symmetric Channel with error rate ϵ. Since the colluders do not control the value of ϵ, their collusion strategy is independent of this additional degradation. Its impact transforms $\boldsymbol{\theta}$ into $\boldsymbol{\theta}(\epsilon) = (1 - 2\epsilon)\boldsymbol{\theta} + \epsilon$.

3 The Worst Case Attack

3.1 Tardos decoding

The accusation proposed by G. Tardos belongs to the class of "simple decoders", using the nomenclature introduced by P. Moulin [7]. For any user j, a simple decoder calculates a score depending on the code sequence $\mathbf{x}_j$, the sequence $\mathbf{y}$

decoded from the pirated copy, and the secret of the code $\mathbf{p} = (p_i, \dots, p_m)$. B. Skoric *et al.* [9] proposed a symmetric version of the original Tardos score:

$$s(\mathbf{x}_j, \mathbf{y}, \mathbf{p}) = \sum_{i \in [m]} \delta_{y(i)}(x(j,i)) \sqrt{\frac{1 - p_{y(i)}}{p_{y(i)}}} - (1 - \delta_{y(i)}(x(j,i))) \sqrt{\frac{p_{y(i)}}{1 - p_{y(i)}}}, \quad (1)$$

with $p_{y(i)} = p_i^{y(i)}(1 - p_i)^{1 - y(i)}$, and $\delta_a(b)$ the Kronecker mapping equalling 1 if $a = b$, 0 else. User j is accused if $s(\mathbf{x}_j, \mathbf{y}, \mathbf{p})$ is bigger than a threshold τ. This last parameter sets the trade-off between probabilities ϵ_2 and ϵ_1.

Paper [5] explains what seems to have been the rationale of G. Tardos. The collusion attack $\boldsymbol{\theta}$ is a nuisance parameter because it is not known at the accusation side. Nevertheless, the performances of the code should be guaranteed whatever the value of this nuisance parameter. The decoding proposed by Tardos and improved by Skoric *et al.* has indeed a very strong invariance property: for a given collusion size, the mean and the variance of the scores of the innocent and the colluders do not depend on $\boldsymbol{\theta}$. Therefore, no collusion is worse than another.

Yet, the rationale of [5] is in practice flawed. The typical behavior of the scores boils down to mean and variance if and only if they are Gaussian distributed. Being the sum of statistically independent random variables, the Central Limit Theorem (CLT) states that this is the case only when the code length is infinite which is of course not true in practice. The achievable rate reflects this fact.

3.2 The Achievable Rate

The rate of a traitor tracing code is defined by $R = \log_2(n)/m$. Loosely speaking, the achievable rate is a parameter which tells the maximum code rate that can be achieved yet guaranteeing a reliable accusation process exists [7]. The achievable rate for the simple decoder against a given collusion attack, under the assumptions stated in this paper, is given by [7]:

$$\begin{aligned} R_{simple}(\boldsymbol{\theta}) &= \mathbb{E}_P\left[I(Y;X|P=p)\right] \\ &= \mathbb{E}_P\left[H(Y|P=p)\right] - \mathbb{E}_P\left[H(Y|X,P=p)\right], \\ &= \mathbb{E}_P\left[D_{KL}(p_{Y|X} \cdot p_X || p_X \cdot p_Y | P)\right], \end{aligned} \quad (2)$$

where $\mathbb{E}_P\left[\right]$ denotes expectation over P, $I(Y;X|P=p)$ is the mutual information between Y and X conditioned on $P = p$, and $H(x) = -x\log_2(x) - (1-x)\log_2(1-x)$ is the binary entropy function [4]. Notice that the achievable rate depends on the considered collusion attack $\boldsymbol{\theta}$ through the probabilities involved in the computation of the mutual information. The equality (2) provides us with an interesting interpretation. The accusation can be seen as a hypothesis test: $\mathcal{H}_0$ the user is innocent, $\mathcal{H}_1$ the user is guilty. The performances of the test are theoretically limited by the Kullback-Leibler distance, D_{KL}, between the pdf of the observations under both hypotheses. Under $\mathcal{H}_0$, the sequence $\mathbf{Y}$ is created from sequences statistically independent from the user sequence $\mathbf{X}$. Therefore, their joint pdf is indeed the product $p_{\mathbf{X}} \cdot p_{\mathbf{Y}}$. Under $\mathcal{H}_1$, on the contrary, there exists a conditional pdf linking the two sequences. The conditioning over P comes from the fact that the accusation uses this secret parameter as side information. The bits of the sequences being independent, the KL distance of the sequences is the sum of the KL distances of the samples, which is m times the expectation per symbol.

The Stein lemma [4] states that, asymptotically, $\epsilon_1 \rightarrow 2^{-mR_{simple}(\boldsymbol{\theta})}$. Consider $n = 2^{mR}$ with R the rate of the code. Then, asymptotically, the probability of falsely accusing any user is bounded by $n\epsilon_1 < 2^{-m(R_{simple}(\boldsymbol{\theta})-R)}$. This bound has a positive error exponent (i.e., an exponential decrease as the code gets longer) if the rate of the code is lower than the achievable rate: $R < R_{simple}(\boldsymbol{\theta})$. Moreover, this should hold for any collusion attack, and thus, for the Worst Case Attack (WCA) defined as follows:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} R_{simple}(\boldsymbol{\theta}). \tag{3}$$

The WCA is thus defined as the collusion attack $\boldsymbol{\theta}^\star$ minimizing the rate of the code, or equivalently, the asymptotic positive error exponent. To calculate the achievable rate, we need the following expressions of the probabilities induced by $\boldsymbol{\theta}$:

$$\mathrm{Pr}_Y[1|p] = \sum_{k=0}^{c} \theta_k \tbinom{c}{k} p^k (1-p)^{(c-k)}, \tag{4}$$

$$\mathrm{Pr}_Y[1|X=1,p] = \sum_{k=1}^{c} \theta_k \tbinom{c-1}{k-1} p^{k-1} (1-p)^{(c-k)}, \tag{5}$$

$$\mathrm{Pr}_Y[1|X=0,p] = \sum_{k=0}^{c-1} \theta_k \tbinom{c-1}{k} p^{k} (1-p)^{(c-k-1)}. \tag{6}$$

The achievable rate is then simply:

$$\begin{aligned} R_{simple}(\boldsymbol{\theta}) &= \mathbb{E}_P \left[H(\mathrm{Pr}_Y[1|p]) \right] \\ &\quad - \mathbb{E}_P \left[p \cdot H(\mathrm{Pr}_Y[1|X=1,p]) - (1-p) \cdot H(\mathrm{Pr}_Y[1|X=0,p]) \right]. \end{aligned} \tag{7}$$

3.3 Identifying the WCA

The expression (7) can be evaluated through numerical integration for a given $\boldsymbol{\theta}$. To identify the WCA, we use a classical optimization algorithm to find the minimum of the achievable rate. This is however only tractable for a small collusion size. Here are the results:

$$\begin{aligned} c = 2 &: \boldsymbol{\theta}^\star = (0, 0.5, 1), \\ c = 3 &: \boldsymbol{\theta}^\star = (0, 0.652, 0.348, 1), \\ c = 4 &: \boldsymbol{\theta}^\star = (0, 0.488, 0.5, 0.512, 1), \\ c = 5 &: \boldsymbol{\theta}^\star = (0, 0.594, 0.000, 1.000, 0.406, 1). \end{aligned} \tag{8}$$

The fact that the WCA satisfies the relationship $\theta_k = 1 - \theta_{c-k}$ for any $k \in [c]$ is very surprising. The WCA belongs to the class of collusion named 'sighted colluders', also known as 'multimedia collusion' [5]. This relationship means that the colluders do not need to know which symbol is indeed in the i-th block of content. They just need to compare their blocks of content and to count the similar versions they received: They have $\Sigma(i)$ times the same version, and $c - \Sigma(i)$ the other version. But, they don't know which version indeed conveys the symbol '1' or '0'. This information is not required for launching the WCA. This is mostly due to the symmetry of the distribution $f(p)$ which implies that symbols '0' and '1' play similar roles.

Another surprise is that there is no connection with the 'extremal strategy' defined by Skoric *et al* [9]. The 'extremal strategy' just focuses on one characteristic of the pdf of the scores (it minimizes the differences between the expectations of the colluder and innocent scores), whereas the WCA minimizes the distance between the two pdf.

4 A Rare Event Analysis

Our algorithm estimating probabilities of rare events is explained in a very general manner, and then its application to traitor tracing is discussed. The goal is to estimate the probability P of the rare event A that a set of data, called hereafter a particle, $\mathbf{x} \in \mathcal{X}$ with a known pdf $p_{\mathbf{X}}$ has a score $s(\mathbf{x})$, with $s(\cdot) : \mathcal{X} \to \mathbb{R}$ a deterministic score function, bigger than a given threshold τ: $\pi = \Pr_{\mathbf{X}}[s(\mathbf{x}) > \tau]$.

4.1 The Key Idea

To factorize a probability into a product of bigger probabilities, we use the following trick: let $A_N = A$ be the rare event, and A_{N-1} a related event such that when A_N occurs, A_{N-1} has also occured. However, when A_{N-1} occurs, it doesn't imply that A_N is true. Hence, A_{N-1} is less rare an event than A_N. This justifies the first equality in the following equation, the second one being just the Bayes rule:

$$\Pr[A_N] = \Pr[A_N, A_{N-1}] = \Pr[A_N|A_{N-1}] \cdot \Pr[A_{N-1}]. \tag{9}$$

Repeating the process, we finally obtain:

$$P = \Pr[A_N] = \Pr[A_N|A_{N-1}]\Pr[A_{N-1}|A_{N-2}] \cdots \Pr[A_2|A_1]\Pr[A_1] \tag{10}$$

provided that $\{A_j\}_{j=1}^N$ is a sequence of nested events. Knowing that estimation of a probability is easier when its value is bigger, we have succeeded in decomposing a hard problem into N much easier problems.

This decomposition is very general, but the construction of this sequence of nested events is usually not a simple task. An exception is when the rare event A_N admits a geometrical interpretation: A_N occurs when $\mathbf{x} \in \mathcal{A}_N$. A sequence of nested events translates then in a sequence of subsets $\mathcal{A}_N \subset \mathcal{A}_N \subset \ldots \subset \mathcal{A}_1$.

The task is even simpler in traitor tracing problem because an indicator function of these events can be as follows: $\mathbf{x} \in \mathcal{A}_j$ if $s(\mathbf{x}) > \tau_j$. Nested events are created for a sequence of increasing thresholds: $\tau_1 < \tau_2 < \cdots < \tau_N = \tau$.

4.2 Generation of vectors

The first step estimates $\pi_1 = \Pr[A_1]$. In practice, N is large enough so that this probability is not lower than 0.1. Then, a classical Monte Carlo estimator is efficient. We generate l_1 vectors $\mathbf{x}$ distributed as $p_\mathbf{x}$, we calculate their score and count the number k_1 of times it is higher than τ_1. This first step leads not only to an estimator $\hat{\pi_1} = k_1/l_1$, but also to a generator of the event A_1. It is not very efficient because approximately only $\pi_1 l_1$ occurrences of the event A_1 are produced.

4.3 Replication of particles

The issue of the second step is the estimation of $\pi_2 = \text{Prob}[A_2|A_1]$. We set the threshold τ_2 just above τ_1, so that this probability is large (typically not lower than 0.1). A MC estimator is $\hat{\pi_2} = k_2/l_2$, where k_2 is the number of particles $\mathbf{x}$ of the set $\mathcal{A}_1$ which also belong to $\mathcal{A}_2$. We need to generate l_2 particles $\mathbf{x}$ distributed according to $p_\mathbf{X}$ and in the set $\mathcal{A}_1$. We could apply the first step of the algorithm to generate these particles, but it is not efficient enough as such.

We then resort to a so-called replication process, which almost multiplies by a factor ρ the size of a collection in a particular region of the space. For each particle, we make ρ copies of it, and then we slightly modify the copies in a random manner. This modification process must leave the distribution of the particle invariant: if its inputs are distributed according to $p_\mathbf{X}$, its outputs must follow the same distribution. A modified copy is likely to belong to the set if the modification is small. However, we check whether this is really the case, and go back to the original particle if not. The replication process is thus a modification followed by a filter.

Since we have run the first step, we have already k_1 particles in $\mathcal{A}_1$. We choose a replication factor $\rho_1 \approx \hat{\pi_1}^{-1}$ approximately keeping the same number of particles. We calculate the scores of the $\rho_1 k_1$ modified copies. We keep the copies whose score is bigger than τ_1, and replace the others by their original particle. This makes the $l_2 = \rho_1 k_1$ input particles of the MC estimator. These two first steps lead to an estimator of π_2 and a generator of events A_2.

The core of the algorithm is thus the following one. The selection kills the particles whose score is lower than an intermediate threshold τ_i, these are branched on selected particles. The replication proposes randomly modified particles and filters those that are still above the intermediate threshold. Selection and replication steps are iterated to estimate the remaining probabilities $\pi_3, \cdots, \pi_N$. The estimator $\hat{\pi}$ is then simply $\hat{\pi_1} \ldots \hat{\pi_N}$.

4.4 Adaptive thresholds

The difficulty is now to give the appropriate values to the thresholds $\{\tau_i\}_1^{N-1}$, and also to the sizes of the sets $\{l_i\}_1^N$. The probabilities to be estimated must not be very weak in order to maintain reasonable set sizes. Moreover, it can be shown that the variance of $\hat{\pi}$ is minimized when the probabilities $\{\hat{\pi}_i\}_i^N$ are equal [6]. We would need the map $\tau = F^{-1}(\pi)$ to set the correct value of the thresholds, which we have not. Otherwise, we would already know the value of $\pi = F(\tau)$.

The idea is to set the thresholds adaptively. The number of particles is kept constant in all the experimental rounds: $l_i = l \, \forall i \in \{1 \cdots N\}$. The threshold τ_i has the value such that $k_i = k$. Thus, k and l are the parameters of the algorithm. The estimated probabilities are all equal to $\pi_i = k/l$, $\forall i \in [N-1]$. It means that the selection sorts the scores in a decreasing order, and adaptively sets τ_i as the value of the k-th higher score. Particles whose score is below this threshold are removed from the stack, and replaced by copies of other particles. The size of the stack is constant and the replication factors $\{\rho_i\}$ are all equal to l/k (k divides l). All the particles in the stack are independently modified. The modification of a particle is accepted only if its new score is still above the threshold.

The last step is reached when $\tau_i > \tau$. Then, we set $N = i$, $\tau_N = \tau$ and k_N is the number of scores above τ, so that, for this last iteration, $\hat{\pi_N} = k_N/l$. At the end, the probability of the event A_N is estimated by:

$$\hat{\pi} = \frac{k_N k^{N-1}}{l^N}. \tag{11}$$

The number of iterations is expected to be as follows:

$$\mathbb{E}[N] = \lfloor \log \pi^{-1} / \log l/k \rfloor + 1. \tag{12}$$

The total number of detector trials is lN, which has a logarithmic scale with respect to π^{-1}, whereas a classical MC estimator would need at least π^{-1} trials.

The method is given in pseudo-code in Algorithm 1. Note that the thresholds $\{\tau_i\}$ are stored in memory. This is not useful when estimating π, but this gives a nice byproduct for ROC curves: the map $\pi = f(\tau)$ is estimated through the following points: $\{((k/l)^j, \tau_j)\}_{j=1}^{N-1}$. From [3], the method inherits the asymptotic properties of consistency and normality, with equations:

$$\hat{\pi} \xrightarrow[l \to +\infty]{a.s.} \pi \tag{13}$$

$$\sqrt{l}(\hat{\pi} - \pi) \xrightarrow[l \to +\infty]{\mathcal{L}} \mathcal{N}(0, \sigma^2) \tag{14}$$

with

$$\sigma^2 \gtrsim \pi^2 \left((N-1) \left(\frac{l}{k} - 1 \right) + \frac{l}{k_N} - 1 \right) \tag{15}$$

4.5 Estimation of ϵ_1

We use our algorithm to estimate the probability of accusing an innocent. Particles in this framework are now binary code sequences of length m. The `GENERATE` subroutine is given by the construction of the code [11] and the `SCORE` function is given by Eq. (1). One very important fact is that the symbols in a code sequence are statistically independent and distributed according to their own law. The `MODIFY` subroutine is thus very simple: we randomly select a fraction μ of them (parameter μ sets the modification strength), and re-generate them according to their own law: $\text{Pr}_{X_{j,i}}[1] = p_i$. These non deterministic changes leave the distribution of the code sequence invariant.

We generate c code sequences of length m. The collusion strategy is the WCA. Then, we estimate the map $\tau = F^{-1}(\text{Pr}_{\mathbf{X}}[s(\mathbf{x}, \mathbf{y}, \mathbf{p}) > \tau | \mathbf{y}, \mathbf{p}])$ with our algorithm. Indeed, the target threshold is fixed to a very high value so that the algorithm stops after $N_{\max}$ iterations. The obtained mapping is $N_{\max}$ couples $(\hat{\tau_j}, (k/l)^j)$. However, this holds for a special occurrence of $\mathbf{y}$ and $\mathbf{p}$. Therefore, we need to integrate this conditional probability over $\mathbf{y}$ and $\mathbf{p}$. To do so, we run W times the estimator. At each run, the secret key $\mathbf{p}$ is different, and c code sequences are drawn independently and uniformly to forge a new pirated sequence. The j-th threshold is averaged over the W estimates. We estimate the plot mapping ϵ_1 and the threshold τ, for different couples (c, m).

4.6 Estimation of ϵ_2

The second part of the experiment measures probability ϵ_2 of missing all colluders, i.e. the false negative for the 'Detect-one'. A particle is now a set of c Tardos sequences $\{\mathbf{x}_1, \dots, \mathbf{x}_c\}$ and sequence $\mathbf{y}$ forged from these c sequences by the WCA. The `MODIFY` subroutine modifies the c sequences as described above.

The `SCORE` function is more complicated. From a particle, it calculates the c accusation sums. The score of the particle is then their mean or their maximum. Tardos and his followers based their analysis on the mean of the scores of the colluders because this leads to tractable equations. The rationale is that if the mean is below the threshold, then there is at least one colluder whose score is lower than the threshold. However, the probability that the mean is below the threshold τ is a very rough estimate of the probability of false negative ϵ_2. We choose to follow Tardos' choice of the mean to appreciate the refinement by our experimental investigation compared to the constants found by Tardos and his followers via Chernoff bounds. However, we can also set the score of a particle as the maximum of the c accusation sums in order to really estimate ϵ_2 as the probability that none of the c colluders gets caught.

The rest of the second part works like the first part. We are interested in estimating the mapping $\tau = F^{-1}(\text{Prob}(\max_{i \in [c]} s(\mathbf{x}_i, \mathbf{y}, \mathbf{p}) < \tau))$ (max or mean) using our algorithm. The experiment is run W times, and the intermediate thresholds are averaged for a better precision. Then, we plot in the same figure (see Fig. 1) the false positive mapping and the false negative mapping, except that for this latter one, the probability is taken to the power $4/c$ to

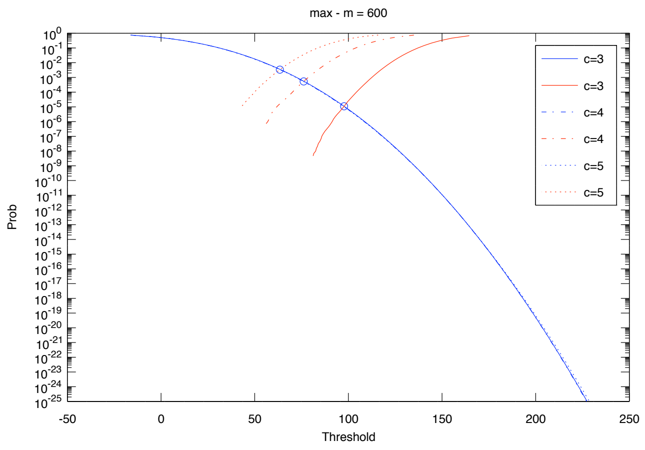

Fig. 1. Mappings of the false positive probability (blue) and false negative probability to the power $4/c$ (red) against the threshold. $m = 600$, $c \in \{3, 4, 5\}$. The score of a particle is the max of the colluders scores. The collusion is the worst as given in (8).

provide a fair comparison to previous evaluations of constant K where ϵ_2 was set to $\epsilon_1^{c/4}$ (original Tardos setup). The intersection of the two mappings at a point $(T_0(m, c), \epsilon_0(m, c))$ implies that it is possible to find a threshold such that $\epsilon_1 = \epsilon_0(m, c)$ and $\epsilon_2 = \epsilon_0(m, c)^{c/4}$. This value indeed reflects the best we can do given m and c. We cannot achieve a lower significance level while preserving the relationship $\epsilon_2 = \epsilon_1^{c/4}$ because no threshold fulfills this constraint at a lower significance level than $\epsilon_0(m, c)$. The only way to get lower significance levels is to increase the code length for a given collusion size.

5 Experimental Evaluation

Several experimentations have been carried out with various code lengths $m \in \{100, 150, 200, 300, 400, 600\}$ and collusion size $c \in \{3, 4, 5\}$ to obtain different values of $\epsilon_0(m, c)$. The final plot draws m against the function $c^2 \log \epsilon_0(m, c)^{-1}$, see Fig. 2.

The most striking fact is that the evaluated lengths are indeed much lower than foreseen by the theoretical bounds in $m > Kc^2 \log \epsilon_1^{-1}$. With the symmetric accusation scoring, $K = 50$ (Tardos) or $K = 2\pi^2$ (Skoric *et al.*). Here, the experiment clearly shows that $m \approx K_1 c^2 \log \epsilon_1^{-1} + K_0(c)$, with a negative $K_0(c)$ and $K_1 < K$. Yet, this is not a contradiction: previous works claimed that if m is bigger than the lower bound, one could be sure that the requirements on the probabilities of errors were met. We just show that indeed these lower bounds are over-pessimistic. It is not very clear whether the slope of the plot K_1 is independent of the collusion size. However, if this holds, this is a very very light dependency, and more experiments should be carried out to confirm this fact.

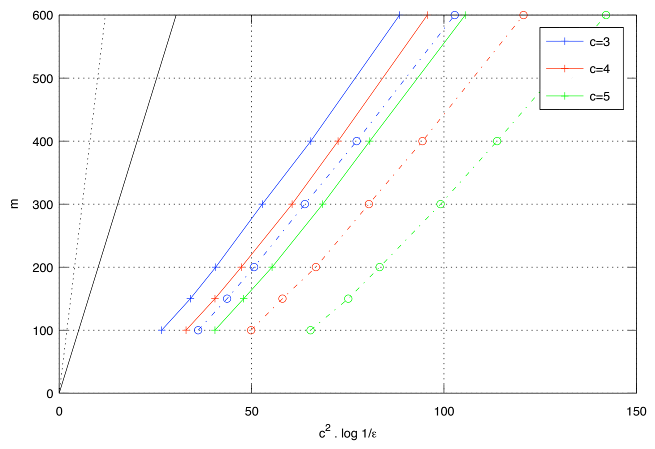

Fig. 2. Code length needed to obtain $\epsilon_2 = \epsilon_1^{c/4}$ for $c = 3$, 4 or 5 colluders (WCA), against function $c^2 \log \epsilon_1^{-1}$. The probability ϵ_2 has been estimated with the mean of the colluders' scores (solid lines) or their maximum (dotted lines). Aforementioned bounds appear in black: Tardos $K = 50$ (dotted line), Skoric *et al.* $K = 2\pi^2$ (solid line).

5.1 Maximum and mean score

The evaluation of the false negative rate from the mean of the scores or their maximum doesn't change the nature of the plot. Whereas constant K_1 is not really different, the difference on constant K_0 might have a big impact in practice: for $c = 4$ and $m = 600$, the minimum achievable error probability is $\epsilon_1 = \epsilon_2 = 2,5 \cdot 10^{-3}$ for the mean score, and $5,3 \cdot 10^{-4}$ for the maximum score, or, in the other way around, for a given ϵ_1 and $c = 4$, the code length is more than 200 bits shorter than foreseen by mean-based estimations. Indeed, these are overestimating the power of the colluders. It is much more difficult to forge a pirated copy such that all the accusation scores is below a given τ, than to forge a pirated copy such that the mean of these scores is below τ.

5.2 The impact of the Worst Collusion Attack

Fig. 3 compares the impact on the code length of the WCA and the Majority attack. Clearly, the WCA increases the length of code for given collusion size and probabilities of errors. A linear predictor shows that the constant K_1 for the WCA is approximately the same as when the collusion attack is a majority vote (cf. Table 1). This supports the commonly accepted fact, as stated in previous articles, that the collusion attack has no impact on the asymptotic expression of the code length when the accusation rule (1) is used. Nevertheless, the WCA imposes a bigger offset K_0, making a big difference for not so small error probabilities.

	Majority vote		WCA		WCA $\epsilon = 0.05$		WCA $\epsilon = 0.1$	
	K_1	K_0	K_1	K_0	K_1	K_0	K_1	K_0
$c = 3$	7.2	-180	7.6	-180	9.3	-210	11.6	-245
$c = 4$	7.0	-270	7.1	-265	8.6	-305	10.6	-355
$c = 5$	6.7	-380	6.6	-340	8.0	-405	9.6	-465

Table 1. Identification of the model $m = K_1 c^2 \log \epsilon_1^{-1} + K_0$.

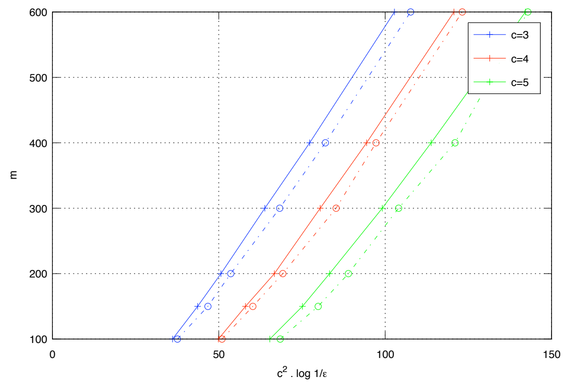

Fig. 3. Code length needed to obtain $\epsilon_2 = \epsilon_1^{c/4}$ for $c = 3$, 4 or 5 colluders. The colluders lead a WCA (solid lines) or a Majority vote (dashed lines).

5.3 The impact of the Binary Symmetric Channel

Fig. 4 shows the impact of the error rate ϵ of the BSC on top of the WCA attack. It is not surprising to note that the code length grows as ϵ increases. Table 1 clearly reveals that the BSC error rate has a dramatic impact on both constants K_1 and K_0. To outline the impact of the BSC, let us consider a channel producing erasures (unreadable bits) with a rate η. At the accusation side, the erasures are not considered, and everything acts as if the code length is reduced by a factor η. Hence, for a given couple (c, ϵ_1), one needs $m\eta$ extra bits, which changes the constants in $K_1(1+\eta)$ and $K_0(1+\eta)$. The impact of the BSC channel is much stronger: an error rate of 5% produces an increase of 20%, an error rate of 10% increases the constant K_1 by 50%. Therefore, it is of utmost importance to resort to a very robust watermarking scheme.

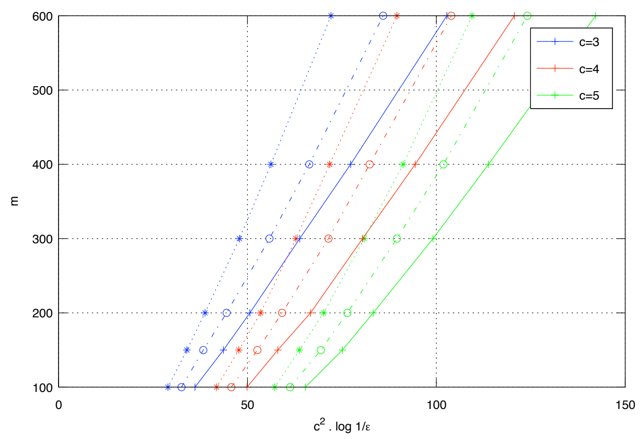

Fig. 4. Code length needed to obtain $\epsilon_2 = \epsilon_1^{c/4}$ for $c = 3$, 4 or 5 colluders. A Binary Symmetric Channel modifies the pirated sequence forged by the WCA, with error rate $\epsilon = 0$ (solid lines), $\epsilon = 0.05$ (dashed lines) and $\epsilon = 0.1$ (dotted lines).

6 Conclusion

The estimation of the code length of a Tardos code requires two skills: to assess what is the worst attack the colluders can lead, and to experimentally assess the probabilities of false positive and false negative for a given collusion size and code length. The worst case attack is defined as the collusion minimizing the achievable rate of the code. This theoretical definition has a very broad scope: whatever the accusation algorithm, we are sure that the colluders cannot lead a stronger attack than this. We propose an estimator of weak probabilities whose scope is far broader than the traitor tracing problem. Our experimental evaluation gives lengths of code which are far smaller than the previously known theoretical lower bounds.

References

1. Oded Blayer and Tamir Tassa. Improved versions of Tardos' fingerprinting scheme. *Des. Codes Cryptography*, 48(1):79–103, 2008.
2. D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Trans. Inform. Theory*, 44:1897–1905, September 1998.
3. F. Cérou, T. Furon, and A. Guyader. Experimental assessment of the reliability for watermarking and fingerprinting schemes. *EURASIP Jounal on Information Security*, ID 414962(ID 414962):12 pages, 2008.
4. T. Cover and J. Thomas. *Elements of information theory*. Number ISBN-0-471-06259-6 in Wiley series in telecommunications. Wiley, 1991.
5. T. Furon, A. Guyader, and F. Cérou. On the design and optimisation of tardos probabilistic fingerprinting codes. In *Proc. of the 10th Information Hiding Workshop*, LNCS, Santa Barbara, Cal, USA, may 2008.

Algorithm 1 Estimation of the probability that $s(\mathbf{x}) > \tau$, when $\mathbf{x} \sim p_{\mathbf{X}}$.

```
Require: subroutines GENERATE, SCORE, HIGHER_SCORE & MODIFY
1: for $i = 1$ to $l$ do
2:   $\mathbf{x}_i = \mathtt{GENERATE}(p_{\mathbf{X}})$; $sx_i = \mathtt{SCORE}(\mathbf{x}_i)$;
3: end for
4: $N = 1$;
5: $\tau_N = \mathtt{HIGHER\_SCORE}(\mathbf{sx}, k)$; $\tau' = \tau_N$;
6: while $\tau' < \tau$ and $N < N_{\mathtt{max}}$ do
7:   $t = 1$;
8:   for $i = 1$ to $l$ do
9:     if $sx_i \geq \tau'$ then
10:       $\mathbf{y}_t = \mathbf{x}_i$; $sy_t = sx_i$; $t = t + 1$;
11:     end if
12:   end for
13:   for $i = 1$ to $k$ do
14:     for $j = 1$ to $l/k$ do
15:       $\mathbf{z} = \mathtt{MODIFY}(\mathbf{y}_i)$;
16:       if $\mathtt{SCORE}(\mathbf{z}) > \tau'$ then
17:         $\mathbf{x}_{(i-1)l/k+j} = \mathbf{z}$; $sx_{(i-1)l/k+j} = \mathtt{SCORE}(\mathbf{z})$;
18:       else
19:         $\mathbf{x}_{(i-1)l/k+j} = \mathbf{y}_i$; $sx_{(i-1)l/k+j} = sy_i$;
20:       end if
21:     end for
22:   end for
23:   $N = N + 1$; $\tau_N = \mathtt{HIGHER\_SCORE}(\mathbf{sx}, k)$; $\tau' = \tau_N$;
24: end while
25: $k' = 0$;
26: for $i = 1$ to $l$ do
27:   if $sx_i > \tau$ then
28:     $k' = k' + 1$;
29:   end if
30: end for
31: return $\hat{\pi} = \frac{k' k^{N-1}}{l^N}$;
```

6. A. Lagnoux. Rare event simulation. *PEIS*, 20(1):45–66, jan 2006.
7. Pierre Moulin. Universal fingerprinting: Capacity and random-coding exponents. *CoRR*, abs/0801.3837, 2008.
8. K. Nuida. An improvement of short 2-secure fingerprint codes strongly avoiding false-positive. In *to appear in proc. of 11th Information Hiding workshop*, 2009.
9. B. Skoric, S. Katzenbeisser, and M. Celik. Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes. *Designs, Codes and Cryptography*, 46(2):137–166, February 2008.
10. B. Skoric, T. Vladimirova, M. Celik, and J. Talstra. Tardos fingerprinting is better than we thought. *IEEE Tran. on IT*, 54(8), August 2008. arXiv:cs/0607131v1.
11. G. Tardos. Optimal probabilistic fingerprint codes. In *Proc. of the 35th annual ACM symposium on theory of computing*, pages 116–125, San Diego, CA, USA, 2003. ACM.