HAL
archives-ouvertes.fr

Decoding Fingerprinting Using the Markov Chain Monte Carlo Method

Teddy Furon, Arnaud Guyader, Frédéric Cérou

To cite this version:

Teddy Furon, Arnaud Guyader, Frédéric Cérou. Decoding Fingerprinting Using the Markov Chain Monte Carlo Method. WIFS - IEEE Workshop on Information Forensics and Security, Dec 2012, Tenerife, Spain. IEEE, 2012. <hal-00757152>

HAL Id: hal-00757152

https://hal.inria.fr/hal-00757152

Submitted on 26 Nov 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Decoding Fingerprints Using the Markov Chain Monte Carlo Method

Teddy Furon [#], Arnaud Guyader [+], Frédéric Cérou [*]

INRIA Rennes [#+], IRMAR [*+], Université de Rennes II [+]*
Rennes, France
teddy.furon@inria.fr

***Abstract*— This paper proposes a new fingerprinting decoder based on the Markov Chain Monte Carlo (MCMC) method. A Gibbs sampler generates groups of users according to the posterior probability that these users could have forged the sequence extracted from the pirated content. The marginal probability that a given user pertains to the collusion is then estimated by a Monte Carlo method. The users having the biggest empirical marginal probabilities are accused. This MCMC method can decode any type of fingerprinting codes.**

This paper is in the spirit of the '*Learn and Match*' decoding strategy: it assumes that the collusion attack belongs to a family of models. The Expectation-Maximization algorithm estimates the parameters of the collusion model from the extracted sequence. This part of the algorithm is described for the binary Tardos code and with the exploitation of the soft outputs of the watermarking decoder.

The experimental body considers some extreme setups where the fingerprinting code lengths are very small. It reveals that the weak link of our approach is the estimation part. This is a clear warning to the '*Learn and Match*' decoding strategy.

I. Introduction

A. The Application

This paper deals with active fingerprinting, a.k.a. traitor tracing. A robust watermarking technique embeds the user's codeword into the content to be distributed. When a pirated copy of the content is scouted, the watermark decoder extracts the message, which identifies the dishonest user. However, there might exist a group of dishonest users, so called collusion, who mix their personal versions of the content to forge the pirated copy. The extracted message no longer corresponds to the codeword of one user, but is a mix of several codewords. The decoder aims at finding back some of these codewords to identify the colluders, while avoiding accusing innocent users.

A popular construction of the codewords is the Tardos fingerprinting code [1]. The recent literature on this type of binary probabilistic code aims at improving either the coding or the decoding side. The first topic uses theoretical arguments to fine-tune the code construction [2], [3]. Our paper has no contribution on this side: we use the classical code construction originally proposed by Tardos in [1] for benchmarking, but our scheme readily works with these more recent codes. As for the decoding side, we pinpoint the following features in the literature: *Hard* decoders work on the binary sequence extracted from the pirated content, whereas *soft* decoders uses some real outputs from the watermarking layer [4]. This paper pertains to this latter trend. Some decoders compute a score per user, which is provably good whatever the collusion strategy, whereas others algorithms first estimate the collusion attack and adapt their scoring function. This paper pertains to this '*Learn and Match*' trend [5]. Last but not least, information theory tells us that a *joint* decoder considering groups of users is potentially more powerful than a *single* decoder computing a score per user. Our approach is based on joint decoding [3].

B. The problem with joint decoding

Very few papers put into practice the principle of joint decoding. The main difficulty is the matter of complexity. If we consider groups of e people among n users, then we need to browse $\binom{n}{e} = O(n^e)$ such groups, which is not tractable for a large database of users. The approach first proposed in [6] for pair decoding and generalized for bigger subsets in [5] is an iterative decoder. The iteration t considers subsets of t users, which are taken from a small pool of most likely colluders of size n_t. If $n_t = O(n^{1/t})$, then the number of subsets to be analyzed remains in $O(n)$ and supposedly within computational resources. But, this implies that more and more users are discarded and this pruning is prone to miss colluders.

Another point is that the scoring function used in [5] is not optimal for three reasons. The scoring is based on the likelihood that a group of users are all guilty, whereas it should be the likelihood that some of them are guilty. The scoring relies on the way the codewords were generated and is extremely specific to the probabilistic nature of the Tardos code construction. It is also based on a pessimistic estimation of the collusion attack artificially assuming a bigger collusion size c_{max}. It appears that this scoring is less discriminative if c_{max} is much larger than the true collusion size c.

C. Our contributions

This paper presents a *soft* decoder in the trend of the '*Learn and Match*' approach. The estimation part elegantly uses the E.-M. algorithm for a fast estimation of the collusion attack.

WIFS'2012, December, 2-5, 2012, Tenerife, Spain. 978-1-4244-9080-6/10/$26.00 *©2012 IEEE.*

After this first task, our decoding algorithm puts into practice the concept of *joint* decoding more directly than the previous methods [6], [5]. Specifically, there is no iterative pruning of the users. Instead of computing scores per user subset, the decoder generates typical subsets likely to be the real collusion. This is efficiently done by a Markov chain and a convenient representation of collusion. A Monte Carlo method computes statistics from these sampled subsets such as the marginals of being a colluder. This Markov Chain Monte Carlo (MCMC) method is the core of our algorithm and is the main contribution of this paper. It works with any code construction (probabilistic or error correction based). This approach has been also used in biology for library screening [7] and in blind deconvolution [8].

Our main contributions are introduced first: the representation of collusion (Sect. II) and the MCMC decoder (Sect. III). The more technical details about the experimental setup (collusion model for the watermarking layer, estimation of its parameters with the E.-M. method, and the derivation of transition probabilities) are presented in the second part of the paper (Sect. IV and V). An experimental investigation shows state-of-the-art performances as well as the limits of our approach with very short codes.

II. The representation of collusion

The keystone of our approach is the representation of collusion by a limited list of the colluder identities. Suppose there are n users indexed from 1 to n and denote $[n] \triangleq \{1, \ldots, n\}$. Depending on how short is the code, tracing collusions bigger than a given size, denoted by c_{max}, produces unreliable decision . The collusion representation is a vector $\mathbf{s}$ of c_{max} integer components, each ranging from 0 to n. Some of them may take the value 0 which codes 'nobody'. We denote $s_0 \triangleq \|\mathbf{s}\|_0$ its ℓ_0-norm, i.e. the number of non-zero components. This vector represents a collusion of size s_0 whose users are given by the non-zero components. Hence, there is no pair of non-zero components with the same value. We denote by $\mathcal{S}$ the set of all such vectors.

For example, with $c_{\mathsf{max}} = 5$, $\tilde{\mathbf{s}} = (6, 0, 3, 2, 0)$ represents the collusion of the $\tilde{s}_0 = 3$ following users: 2, 3, and 6.

A. The Neighborhood of a collusion

We denote by $\mathcal{S}(\mathbf{s})$ the neighborhood of $\mathbf{s}$ as the set of collusions differing by at most one component in their representation. We have $\mathbf{s} \in \mathcal{S}(\mathbf{s})$ and the other neighbors have one more colluder, one less colluder, or just one different colluder. For instance, $\{(6, 1, 3, 2, 0), (6, 0, 3, 0, 0), (6, 0, 4, 2, 0)\} \subset \mathcal{S}(\tilde{\mathbf{s}})$.

The neighborhood is decomposed as $\mathcal{S}(\mathbf{s}) = \bigcup_{i=1}^{c_{\mathsf{max}}} \mathcal{S}_i(\mathbf{s})$, with

$$\mathcal{S}_i(\mathbf{s}) \triangleq \{\mathbf{s}' \in \mathcal{S} | s'(k) = s(k), \forall k \neq i\}. \tag{1}$$

The subsets $\mathcal{S}_i(\mathbf{s})$ are not disjoint. If $s(i) = 0$, then $\mathcal{S}_i(\mathbf{s})$ is composed of $\mathbf{s}$ and some collusions of size $s_0 + 1$ (one user is added). The cardinal of $\mathcal{S}_i(\mathbf{s})$ equals $n + 1 - s_0$. If $s(i) > 0$, then $\mathcal{S}_i(\mathbf{s})$ is composed of some collusions of size s_0 including $\mathbf{s}$ (user $s(i)$ is replaced by someone else or not) and one collusion of size $s_0 - 1$ (user $s(i)$ is removed). In this case, $|\mathcal{S}_i(\mathbf{s})| = n + 2 - s_0$.

B. The prior probability of a collusion

We now cast a probabilistic model on the collusion representation. Our prior is as less informative as possible. Having no information about the size of the collusion, we pose that all sizes are equally probable if less or equal than c_{max}:

$$\mathbb{P}(s_0) = c_{\mathsf{max}}^{-1}, \forall s_0 \in [c_{\mathsf{max}}]. \tag{2}$$

We also pose that the $\binom{n}{c}$ collusions of size c are equally likely. Finally, the prior on $\mathbf{s}$ is given by:

$$\mathbb{P}(\mathbf{s}) = \mathbb{P}(\mathbf{s}, s_0) = \mathbb{P}(\mathbf{s}|s_0)\mathbb{P}(s_0) = \binom{n}{s_0}^{-1} c_{\mathsf{max}}^{-1}. \tag{3}$$

With this model, the prior distribution is not uniform: collusions of bigger size have lower probabilities.

C. The posterior probability of a collusion

Once a pirated copy is scouted, a sequence $\mathbf{z}$ is extracted. This observation refines our knowledge on the collusion, which is reflected by the posterior probability $\mathbb{P}(\mathbf{s}|\mathbf{z})$. Thanks to the Bayes rule:

$$\mathbb{P}(\mathbf{s}|\mathbf{z}) = \frac{\mathbb{P}(\mathbf{z}|\mathbf{s})\mathbb{P}(\mathbf{s})}{\mathbb{P}(\mathbf{z})}. \tag{4}$$

Yet, we cannot compute this probability because $\mathbb{P}(\mathbf{z})$ is unknown. This is not critical since this quantity is a common denominator to all posteriors: we can still compare posteriors or compute the ratio of two posteriors. The next difficulty is the conditional $\mathbb{P}(\mathbf{z}|\mathbf{s})$, which in words is the probability that collusion $\mathbf{s}$ could have forged sequence $\mathbf{z}$. This is where we need a probabilistic model of the collusion process. Sect. IV-C presents our models and Sect. V-D gives the expressions of the conditional probabilities.

The Maximum A Posteriori decoder consists in finding the collusion with the biggest posterior. Browsing all of them is yet computationally intractable for a large database of users.

As an alternative, we can build a single decoder based on the marginal posterior probabilities: the probability that user j is a colluder is given by

$$\mathbb{P}(j|\mathbf{z}) = \sum_{\mathbf{s}|\exists s(i)=j} \mathbb{P}(\mathbf{s}|\mathbf{z}). \tag{5}$$

Again, browsing the collusions of all the summands is not practical for a large number of users.

III. The Markov Chain Monte Carlo Method

The key idea of our approach is to estimate the above marginals with a Markov Chain Monte Carlo method.

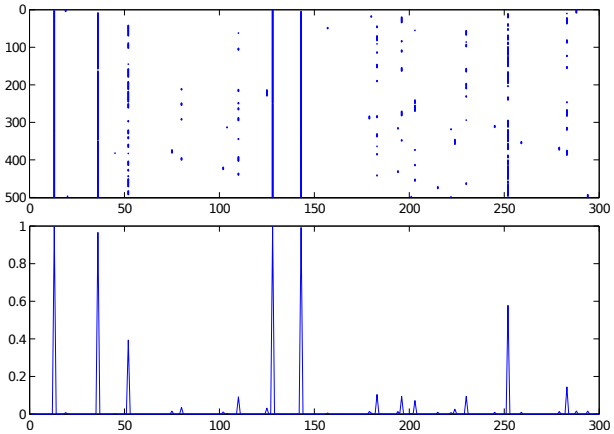

Fig. 1. Illustration of the MCMC metod for $K = 500$ and $n = 300$: [Up] The Markov chain: the binary matrix $K \times n$ indicating which users belong to the state $\mathbf{s}^{(t)}$ for $1 \leq t \leq K$, [Down] The Monte Carlo estimation: the empirical marginal probabilities, i.e. the mean of the columns of the above binary matrix.

A. The Monte Carlo Method

Instead of computing the marginals with (5), a Monte Carlo method estimates their values: we draw K collusions $\{\mathbf{s}_k\}_{k=1}^K$ according to distribution $\mathbb{P}(\mathbf{s}|\mathbf{z})$, and the empirical marginals are given by:

$$\hat{\mathbb{P}}(j|\mathbf{z}) = |\{\mathbf{s}_k | \exists \mathbf{s}_k(i) = j\}|/K, \tag{6}$$

which reads as the empirical frequency that user j belongs to these sampled collusions. The next subsections explain how we succeed to sample collusions distributed as $\mathbb{P}(\mathbf{s}|\mathbf{z})$. The difficulty lies in the fact that we cannot sample them directly because we only know $\mathbb{P}(\mathbf{s}|\mathbf{z})$ up to the multiplicative constant $\mathbb{P}(\mathbf{z})$.

B. The Markov Chain

The collusions are indeed generated thanks to a Markov chain. It is an iterative process with a state (here a collusion) taking value $\mathbf{s}^{(t)}$ at iteration t. The next iteration draws a new state $\mathbf{s}^{(t+1)}$ according to the transition probability $\mathbb{P}(\mathbf{s}^{(t+1)} = \mathbf{s}|\mathbf{s}^{(t)})$ specified in the next section. The Markov chain is initialized by randomly selecting a collusion $\mathbf{s}^{(0)}$. The transition probabilities are carefully crafted so that the distribution of the state $\mathbf{s}^{(t)}$ converges to the targeted distribution $\mathbb{P}(\mathbf{s}|\mathbf{z})$ as t increases. After a *burn-in* period T, we assume that the Markov chain has forgotten $\mathbf{s}^{(0)}$ and that the states are correctly sampled from now on: the collusions $\{\mathbf{s}^{(t)}\}_{t=T}^{T+K}$ are then passed to the Monte Carlo part of the algorithm that computes the empirical marginals thanks to (6). Fig. 1 illustrates the MCMC method. The colluders are the 6 users with the highest empirical marginals. One sees that innocents often belong to some collusion states $\mathbf{s}^{(t)}$, but it intermittently occurs for one given innocent so that eventually it doesn't harm.

C. The Gibbs sampler

Instead of computing the transition probabilities from $\mathbf{s}^{(t)}$ to any possible collusion, we restrict the transitions to the collusions of a subset of its neighborhood $\mathcal{S}(\mathbf{s}^{(t)})$ (See Sect. II-A). This is called a multi-stage Gibbs sampler with random scan [9, Alg. A.42]. At iteration $t+1$, an integer i is first uniformly drawn in $[c_{\mathsf{max}}]$ that indicates the subset $\mathcal{S}_i(\mathbf{s}^{(t)})$. Then, the following transition distribution is constructed:

$\forall \mathbf{s} \in \mathcal{S}_i(\mathbf{s}^{(t)})$

$$\begin{aligned} \mathbb{P}(\mathbf{s}^{(t+1)} = \mathbf{s}|\mathbf{s}^{(t)}) &:= \frac{\mathbb{P}(\mathbf{s}|\mathbf{z})}{\sum_{\mathbf{s}' \in \mathcal{S}_i(\mathbf{s}^{(t)})} \mathbb{P}(\mathbf{s}'|\mathbf{z})} \\ &= \frac{\mathbb{P}(\mathbf{z}|\mathbf{s})\mathbb{P}(\mathbf{s})}{\sum_{\mathbf{s}' \in \mathcal{S}_i(\mathbf{s}^{(t)})} \mathbb{P}(\mathbf{z}|\mathbf{s}')\mathbb{P}(\mathbf{s}')} \end{aligned} \tag{7}$$

This choice guarantees that the stationary distribution of this Markov chain is $\mathbb{P}(\mathbf{s}|\mathbf{z})$, which legitimates our approach [9, Sect. 10.2.1]. The unknown multiplicative constant $\mathbb{P}(\mathbf{z})$ in (4) has disappeared in the ratio. This transition distribution only depends on the priors $\mathbb{P}(\mathbf{s}')$ and $\mathbb{P}(\mathbf{s})$ given by (3), and the conditional probabilities $\{\mathbb{P}(\mathbf{z}|\mathbf{s})\}_{\mathbf{s} \in \mathcal{S}_i(\mathbf{s}^{(t)})}$ which depend on the collusion process that will be estimated (See Sect. IV-C and (17) or (18)). These latter quantities are functions of the codewords of users listed in $\mathbf{s}$, whatever their construction. This decoding algorithm thus works with any fingerprinting code. The remainder of the paper applies this approach to Tardos codes.

IV. The Setup

This section briefly reviews the construction of the Tardos code, describes the setup for hiding the user's codeword into the content, and presents the model of the collusion attack.

A. Tardos Code Construction

The binary code is composed of n codewords of m bits. The codeword $\mathbf{x}_j = (x_{j,1}, \cdots, x_{j,m})^T$ identifying user $j \in \mathcal{U} = [n]$, is composed of m binary symbols independently drawn at the code construction s.t. $\mathbb{P}(x_{j,i} = 1) = p_i$, $\forall (j,i) \in [n] \times [m]$. At initialization, the auxiliary variables $\{p_i\}_{i=1}^m$ are independent and identically drawn according to distribution $f(p) : [0,1] \to \mathbb{R}^+$. This distribution is a parameter to be selected by the code designer. Tardos originally proposed $f_T(p) \triangleq \mathbb{1}_{[t,1-t]}(p)\kappa_t/\sqrt{p(1-p)}$ in [1], where $\mathbb{1}_{[a,b]}$ is the indicator function of the interval (i.e. $\mathbb{1}_{[a,b]}(p) = 1$ if $a \leq p \leq b$, 0 otherwise), and κ_t the constant s.t. $\int_0^1 f_T(p)dp = 1$. The cut-off parameter t is usually set to $1/(300c_{\mathsf{max}})$. The integer c_{max} is the maximum expected number of colluders.

The distribution f is public, but both the code $\Xi = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ and the auxiliary sequence $\mathbf{p} = (p_1, \ldots, p_m)^T$ must be kept as secret parameters. A user possibly knows his codeword, but the only way to gain knowledge on some other codewords is to team up in a collusion.

B. The Modulation

We assume the content is divided into chunks. For instance, a movie is split into blocks of frames. A watermarking technique embeds a binary symbol per chunk, sequentially hiding the codeword into the content. This implies that the code length m is limited by the watermarking embedding rate times the duration (or size) of the content. The robustness of the

watermarking technique strongly depends on the embedding rate: the lower the embedding rate, the more robust the watermark. Therefore, it is crucial to design a fingerprinting decoder able to trace colluders despite a short code length.

The watermarking decoder retrieves the hidden bit x per watermarked chunk of content. We assume that it first computes a statistic z, so-called *soft output*, which ideally equals 1 (resp. -1) if the hidden bit is '1' (resp. '0'). The decoder then threshold the soft output at 0 to yield a hard output: $\hat{x} =$ '1' if $z \geq 0$, '0' otherwise. This is the case for instance with a spread spectrum based watermarking technique using an antipodal modulation (a.k.a. BPSK).

In this paper, the fingerprinting decoder (i.e. the accusation process) is named *soft decoder* because it uses the soft outputs of the watermarking decoder. This brings more information on the colluders' identity than the hard outputs.

C. The collusion Attack

The model of the collusion attack is taken from [5]. The partition of content into chunks is not a secret. We assume that the collusion attack sequentially processes all the chunks in the same way: there is first a fusion of the versions of the colluders into one chunk, and then a process (coarse source compression, noise addition, filtering etc.) further distorts this fused chunk. We foresee the following types of fusion.

1) Attacks of type I: The fusion is a signal processing operation that mixes c chunks of content into one. This includes sample-wise average or median, sample patchwork etc. The fusion aims at reducing the confidence in the decoded symbol so we assume that $|z| \leq 1$.

This mixing is strongly driven by the number k of chunks watermarked with symbol '1' (denoted '1'-chunk). For a collusion of size c, $k \in \{0, \ldots, c\}$ and we assume that z can take $c+1$ values $\{\mu_c(k)\}_{k=0}^{c}$. In the manner of the marking assumption, we restrict the power of the collusion by enforcing $\mu_c(c) = -\mu_c(0) = 1$. In other words, there is no fusion when all chunks are watermarked with the same symbol.

The watermarking secret key prevents the colluders from determining the hidden symbol. Yet, they can group their c chunks into two groups of exactly equal instances[1]. This reveals the number of hidden '1' up to an ambiguity: it is k or $c-k$. This implies a symmetry in the model: $\mu_c(k) = -\mu_c(c-k)$, $\forall k \in \{0, \ldots, c+1\}$.

At last, the distortion on the fused chunk adds a noise on the soft output: $z = \mu_c(k) + n$. We assume that this noise is independent of k, and i.i.d. with $n \sim \mathcal{N}(0, \sigma^2)$.

We give two examples taken from [4].

- 'Average': $\mu_c(k) = 2kc^{-1} - 1$, $\forall k$. This is the case for instance if the soft decision is a linear process and the colluders fuse all their chunks with a sample-wise average.
- 'Average2': $\mu_c(k) = 0$, $\forall k \in [c-1]$ and $\mu_c(c) = -\mu_c(0) = 1$. This is the case for instance when the soft decision is a linear process and the colluders fuse only two different blocks with a sample-wise average.

[1]In other words, the watermarking is deterministic: its result only depends on its inputs (the original chunk, the symbol to be embedded, and the secret key)

2) Attacks of type II: In this type, the colluders benefit from the division into blocks. At a given block index, they copy and paste one of their chunks. There is no fusion of blocks. The probability they put a '1'-chunk when they have k chunks watermarked with symbol '1' over c is denoted by $\theta_c(k)$. The marking assumption imposes that $\theta_c(c) = 1 - \theta_c(0) = 1$. The ambiguity on the number of '1'-chunks results in the symmetry $\theta_c(k) = 1 - \theta_c(c-k)$. After the distortion on the selected chunk, z is distributed as $\mathcal{N}(-1, \sigma^2)$ with probability $1-\theta_c(k)$ or as $\mathcal{N}(1, \sigma^2)$ with probability $\theta_c(k)$.

Here are two classical examples also used in [4]:

- 'Uniform': $\theta_c(k) = kc^{-1}$, $\forall k$. The colluders uniformly draw one of their chunks.
- 'Majority': $\theta_c(k) = 1$ if $k > c/2$, 0 otherwise ($\theta_c(c/2) = 1/2$ if c is even). The colluders classify the chunks into two groups of same instance and choose a chunk from the bigger group.

V. The Estimation with the E.-M. Algorithm

The first task of the proposed decoder is the estimation of the collusion attack. It amounts to guess the type of the attack and its parameters $(\boldsymbol{\mu}_c, \sigma)$ (type I) or $(\boldsymbol{\theta}_c, \sigma)$ (type II) from the observations $\mathbf{z}$ and the knowledge of the secret $\mathbf{p}$. The model is not identifiable if c is unknown. For this reason, the estimation is done for all collusion sizes ranging from 1 to c_{max}. This results in a set of c_{max} model parameters. This estimation task heavily relies on the collusion model (Sect. IV-C) and the Tardos code.

A. The Maximum Likelihood Estimator

The MLE (Maximum Likehood Estimator) searches the maximum of the log-likelihood. For type I, the latter has the following expression:

$$\begin{aligned} L^{(I)}(\boldsymbol{\mu}_{\tilde{c}}, \sigma) &\triangleq \log \mathbb{P}(\mathbf{z}|\mathbf{p}, \boldsymbol{\mu}_{\tilde{c}}, \sigma) \\ &= \sum_{i=1}^{m} \log \left(\sum_{k=0}^{\tilde{c}} \mathbb{P}(k|p_i) f(z_i; \mu_{\tilde{c}}(k), \sigma) \right) \end{aligned} \quad (8)$$

with $f(z; \mu, \sigma) \triangleq e^{-\frac{(z-\mu)^2}{2\sigma^2}} / \sqrt{2\pi\sigma^2}$. For type II, we have

$$\begin{aligned} L^{(II)}(\boldsymbol{\theta}_{\tilde{c}}, \sigma) &\triangleq \log \mathbb{P}(\mathbf{z}|\mathbf{p}, \boldsymbol{\theta}_{\tilde{c}}, \sigma) \\ &= \sum_{i=1}^{m} \log \Big(\sum_{k=0}^{\tilde{c}} \mathbb{P}(k|p_i) [\theta_{\tilde{c}}(k) f(z_i; 1, \sigma) \\ &+ (1 - \theta_{\tilde{c}}(k)) f(z_i; -1, \sigma)] \Big) . \end{aligned} \quad (9)$$

The final decision about the type of the attack is taken by comparing the values $L^{(I)}(\boldsymbol{\mu}^{\star}_{\tilde{c}}, \sigma^{\star}_{I,\tilde{c}})$ and $L^{(II)}(\boldsymbol{\theta}^{\star}_{\tilde{c}}, \sigma^{\star}_{II,\tilde{c}})$: the type giving the biggest likelihood is selected. Note that both types have the same number of parameters, so that none is more prone than the other to overfitting and this justifies the comparison of their respective likelihoods.

Yet, these two optimizations are nothing trivial because the functionals have plenty of local maxima. Under both attack types, the observation $\mathbf{z}$ is indeed a mixture of a fixed

number of Gaussian distributions, a typical case where the Expectation-Maximization estimator is elegant and efficient even if convergence to the global maximum is not ensured. The following two subsections are the application of [10, Tab. 3.1].

B. E.M. Algorithm for type I Attacks

We introduce the unknown latent variables $\boldsymbol{\varphi} = (\varphi_1, \ldots, \varphi_m)$ that capture the number of '1'-chunks per index: $\forall i \in [m], \varphi_i \in \{0, \ldots, \tilde{c}\}$. If $\tilde{c} = c$, the true value of φ_i is $\sum_{k \in \mathcal{C}} x_{k,i}$. The log-likelihood function with these latent variables is now $\mathcal{L}^{(I)}(\boldsymbol{\mu}_{\tilde{c}}, \sigma) \triangleq \log \mathbb{P}(\mathbf{z}, \boldsymbol{\varphi} | \mathbf{p}, \boldsymbol{\mu}_{\tilde{c}}, \sigma)$:

$$\mathcal{L}^{(I)}(\boldsymbol{\mu}_{\tilde{c}}, \sigma) = \sum_{i=1}^{m} \log \left(\mathbb{P}(\varphi_i | p_i) f(z_i; \mu_{\tilde{c}}(\varphi_i), \sigma) \right), \quad (10)$$

with $\mathbb{P}(\varphi|p) = \binom{\tilde{c}}{\varphi} p^{\varphi} (1-p)^{\tilde{c}-\varphi}$. The E.-M. iteratively refines the estimation $(\hat{\boldsymbol{\mu}}_{\tilde{c}}, \hat{\sigma})$ of the parameters and of the $(\tilde{c}+1) \times m$ matrix $\mathbf{T}$ storing the conditional probabilities of $\boldsymbol{\varphi}$:

$$T_{k,i} \triangleq \mathbb{P}(\varphi_i = k | z_i, p_i, \hat{\boldsymbol{\mu}}_{\tilde{c}}, \hat{\sigma}), \quad \forall k \in \{0, \ldots, \tilde{c}\}. \quad (11)$$

1) E-step: Given the current estimate of the model, the conditional probabilities of $\boldsymbol{\varphi}$ are updated via the Bayes rule:

$$\hat{T}_{k,i} = \frac{\mathbb{P}(\varphi_i = k | p_i) f(z_i; \hat{\mu}_{\tilde{c}}(k), \hat{\sigma})}{\sum_{u=0}^{\tilde{c}+1} \mathbb{P}(\varphi_i = u | p_i) f(z_i; \hat{\mu}_{\tilde{c}}(k), \hat{\sigma})} \quad (12)$$

2) M-step: Given the conditional probabilities, this step updates the model by finding the parameters that maximize the Q function $\mathbb{E}_{\boldsymbol{\varphi}|\mathbf{zp}}[\mathcal{L}^{(I)}(\boldsymbol{\mu}_{\tilde{c}}, \sigma)]$. These have a closed-form expression in the case of Gaussian mixtures:

$$\hat{\mu}_{\tilde{c}}(k) = \left(\sum_{i=1}^{m} \hat{T}_{k,i} z_i \right) / \sum_{i=1}^{m} \hat{T}_{k,i}, \ \forall k \in [\tilde{c}-1] \quad (13)$$

$$\hat{\sigma}^2 = m^{-1} \sum_{k=0}^{\tilde{c}} \sum_{i=1}^{m} \hat{T}_{k,i} (z_i - \hat{\mu}_{\tilde{c}}(k))^2 \quad (14)$$

These two steps are iterated until the increase of the true log-likelihood $L^{(I)}(\hat{\boldsymbol{\mu}}_{\tilde{c}}, \hat{\sigma})$ is no longer sensitive.

C. E.M. Algorithm for type II Attacks

Under this type of attack, the latent variable $\zeta_i \in \{-\tilde{c}, \ldots, -1, 0, 1, \ldots, \tilde{c}\}$ captures the event that the colluders have $|\zeta_i|$ '1'-chunks and that they copy-paste a chunk with symbol $\mathrm{sg}(\zeta_i)$ at the i-th index ($\mathrm{sg}(k) = 1$ if $k > 0$, 0 otherwise). The log-likelihood function with these latent variables is now $\mathcal{L}^{(II)}(\boldsymbol{\theta}_{\tilde{c}}, \sigma) \triangleq \log \mathbb{P}(\mathbf{z}, \boldsymbol{\zeta} | \mathbf{p}, \boldsymbol{\theta}_{\tilde{c}}, \sigma)$:

$$\mathcal{L}^{(II)}(\boldsymbol{\theta}_{\tilde{c}}, \sigma) = \sum_{i=1}^{m} \log \left(\pi(\zeta_i, p_i) . f(z_i; 2\mathrm{sg}(\zeta_i) - 1, \sigma) \right), \quad (15)$$

with $\pi(\zeta_i, p_i) = \theta(|\zeta_i|)^{\mathrm{sg}(\zeta_i)} (1 - \theta(|\zeta_i|))^{1-\mathrm{sg}(\zeta_i)} \mathbb{P}(|\zeta_i| | p_i)$, and $\mathbb{P}(|\zeta||p) = \binom{\tilde{c}}{|\zeta|} p^{|\zeta|} (1-p)^{\tilde{c}-|\zeta|}$.

1) E-step: The estimates of the conditional probabilities $U_{k,i} \triangleq \mathbb{P}(\zeta_i = k | z_i, p_i, \hat{\boldsymbol{\theta}}_{\tilde{c}}, \hat{\sigma})$ are updated as follows:

$$\hat{U}_{k,i} \propto \begin{cases} \hat{\theta}(k) \mathbb{P}(k|p_i) f(z_i; 1, \hat{\sigma}) & \text{if } 0 \leq k \leq \tilde{c} \\ (1 - \hat{\theta}(k)) \mathbb{P}(k|p_i) f(z_i; -1, \hat{\sigma}) & \text{if } -\tilde{c} \leq k < 0 \end{cases}$$

such that $\sum_{k=-\tilde{c}}^{\tilde{c}} \hat{U}_{k,i} = 1$.

2) M-step: The estimate of the model is updated as follows:

$$\hat{\theta}(k) = \left(\sum_{i=1}^{m} \hat{U}_{k,i} \right) / \left(\sum_{i=1}^{m} \hat{U}_{k,i} + \hat{U}_{-k,i} \right), \ \forall k \in [\tilde{c}-1]$$

$$\hat{\sigma}^2 = m^{-1} \sum_{i=1}^{m} \sum_{k=-\tilde{c}}^{\tilde{c}} \hat{U}_{k,i} (z_i - 2\mathrm{sg}(k) + 1)^2 \quad (16)$$

D. The expression of the conditional probabilities

Once these c_{max} estimations of the collusion attacks are done, these models are used in the MCMC decoder via the conditional probabilities $\mathbb{P}(\mathbf{z}|\mathbf{s})$. Denote $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_m)$ the sequence of number of symbols '1' in the codewords of collusion $\mathbf{s}$: $\kappa_i = \sum_{j \in \mathbf{s}} x_{j,i}$ (with the convention that $x_{0,i} = 0$). If for the size s_0, a type I collusion attack has been diagnosed, then

$$\mathbb{P}(\mathbf{z}|\mathbf{s}) = \prod_{i=1}^{m} f(z_i; \mu^{\star}_{s_0}(\kappa_i), \sigma^{\star}_{I,s_0}), \quad (17)$$

otherwise, for a type II attack:

$$\begin{aligned} \mathbb{P}(\mathbf{z}|\mathbf{s}) &= \prod_{i=1}^{m} \big(\theta^{\star}_{s_0}(\kappa_i) . f(z_i; 1, \sigma^{\star}_{II,s_0}) \\ &+ (1 - \theta^{\star}_{s_0}(\kappa_i)) . f(z_i; -1, \sigma^{\star}_{II,s_0}) \big) \end{aligned} \quad (18)$$

VI. Experimental Body

The experimental setup is the same as in [4] with 'Uniform', 'Majority', 'Average' and 'Average2' attacks (See examples of Sect. IV-C.1 and IV-C.2). There are two scenarios:

(a) $m = 2048$, $c = 8$, $c_{\mathsf{max}} = 10$,
(b) $m = 1024$, $c = 6$, $c_{\mathsf{max}} = 8$.

The common parameters are $n = 10^4$, $T = 400$, $K = 2000$. The variance of the noise is given by $\sigma = 10^{-\mathsf{SNR}/20}$, with SNR ranging from 10 to 2 dB. This is the signal to noise power ratio at the output of the watermark decoder. It should not be confused with the amount of noise on the content samples.

These settings are very extreme in the sense that the codes are too short to be used in practice. We made this choice in order to show the limits of our method.

The set $\mathcal{A}$ is defined as $\mathcal{A} \triangleq \{j | \hat{\mathbb{P}}(j|\mathbf{z}) > \tau\}$. A false negative occurs if this set is empty. In a single decoder, the user in $\mathcal{A}$ with the biggest empirical marginal is accused. In case of a tie where d users have the maximum score, one user is randomly picked. This leads to a false positive with probability d_i/d where d_i is the number of innocents with this maximum score. In a joint decoder, the users in $\mathcal{A}$ are all accused. We record the number of caught colluders $|\mathcal{A} \cap \mathcal{C}|$ and the number of accused innocents $|\mathcal{A}| - |\mathcal{A} \cap \mathcal{C}|$.

Under scenario (a), the comparison with the performances of the decoder proposed in [4] is mitigated as shown in Fig. 2 (down). The baseline of [4] is slightly better against the 'Majority' attack whereas our decoder is better against the three other attacks. Yet, our decoder has a major drawback: the probability of false alarm is totally unacceptable at low SNR (i.e., at 2 dB in Fig. 2) and when facing the 'Uniform' attack. The same comment holds for the scenario (b) (See Fig. 3).

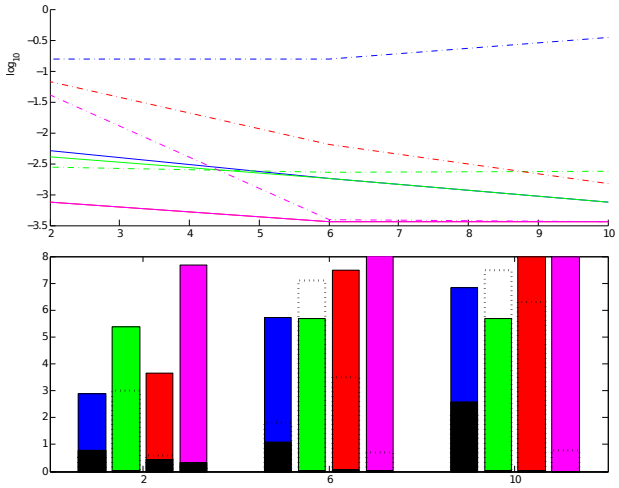

Fig. 2. Scenario (a), 2250 experiments, $\mathsf{SNR} \in \{2, 6, 10\}$ dB. [Up] Probabilities of errors for the single decoder: (solid) Probability of false negative, (dashed) Probability of false positive, [Down] Average number of caught colluders for the joint decoder: (black) average number of accused innocents, (dashed) the baseline from [4], Color: (blue) 'Uniform', (green) 'Majority', (red) 'Average', (pink) 'Average2'.

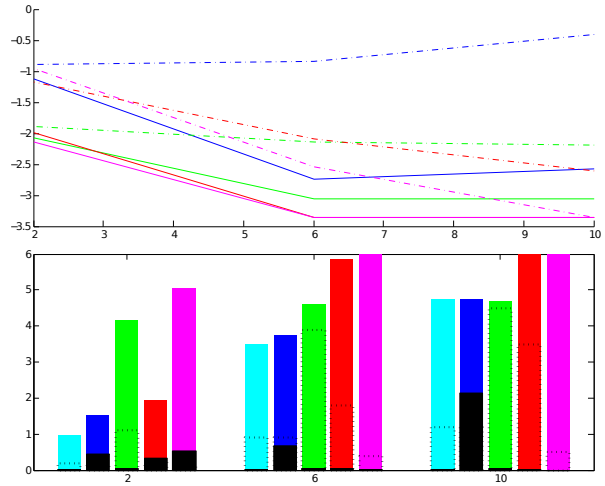

Fig. 3. Scenario (b), 2715 experiments, $\mathsf{SNR} \in \{2, 6, 10\}$ dB. [Up] Probabilities of errors for the single decoder (solid) Probability of false negative - (dashed) Probability of false positive, [Down] Average number of caught colluders for the joint decoder, (black) average number of accused innocents , (dashed) the baseline from [4], Color: (blue) 'Uniform', (green) 'Majority', (red) 'Average', (pink) 'Average2', and (light blue) 'Uniform' attack with true model.

We suspect the 'Uniform' attack to be closed to the worst attack of type I, at a given SNR: it has been proven that the achievable rate against the 'Uniform' quickly converges to the one of the worst case attack when considering hard outputs [3].

The scenario (b) gives us another hint. Both the length of the code and the square of the collusion size are now twice smaller. Since the code length should be of order $m \sim O(c^2)$, the performances of the decoder should be roughly the same under both scenarios. This is not the case: the probability of false accusation is bigger. This allows us to suspect the estimation part to be the Achilles' heel of our algorithm: it produces a bad quality estimation because the code is too short and this spoils the MCMC decoding part. A close look at the estimated collusion models reveals that the type of model and the variance of the noise on the soft outputs are almost always evaluated with high fidelity. Under the 'Uniform' attack, the problem indeed stems from the estimations of $\hat{\boldsymbol{\theta}}_{\tilde{c}}$ with $\tilde{c} > c$ prone to overfitting: these estimated models are quite different in nature from the 'Uniform' attack of size $\tilde{c}$.

To confirm this intuition by a last experiment, we by-pass the estimation and give to the MCMC method the collusion models exactly matching the 'Uniform' attack. This simulates a 'perfect' estimation. The results are shown in Fig. 3 (down) in light blue color. If the number of caught colluders decreases a little, the accusation is now much more reliable with a probability of false positive around 3.10^{-3}.

VII. Conclusion

The fingerprinting decoder studied in this paper shows some novelty: i) the accusation is probabilistic because of the randomness of the MCMC, ii) it resorts to a posteriori probabilities for groups of users more directly than in the previous approach, iii) the decoding part makes no assumption on the code construction. It has also some drawbacks: i) the estimation part detailed here is specific to Tardos fingerprinting codes, ii) the complexity of MCMC is quite high in $O(Kmn)$, iii) the probability of false alarm is not easy to control.

The main message is that the performances are limited by the quality of the estimation of the collusion strategy. This is a clear warning to the 'Learn and Match' decoding strategy, and this issue deserves more research efforts.

References

[1] G. Tardos, "Optimal probabilistic fingerprint codes," in *Proc. of the 35th annual ACM symposium on theory of computing*. San Diego, CA, USA: ACM, 2003, pp. 116–125.

[2] K. Nuida, S. Fujitsu, M. Hagiwara, T.Kitagawa, H. Watanabe, K. Ogawa, and H. Imai, "An improvement of discrete Tardos fingerprinting codes," *Designs, Codes and Cryptography*, vol. 52, no. 3, pp. 339–362, 2009.

[3] Y.-W. Huang and P. Moulin, "On the saddle-point solution and the large-coalition asymptotics of fingerprinting games," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 160–175, 2012.

[4] M. Kuribayashi, "Bias equalizer for binary probabilistic fingerprinting codes," in *Proc. of 14th Information Hiding Conference*, ser. LNCS, S. Verlag, Ed., Berkeley, CA, USA, may 2012.

[5] P. Meerwald and T. Furon, "Towards practical joint decoding of binary Tardos fingerprinting codes," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 4, pp. 1168–1180, August 2012.

[6] E. Amiri, "Fingerprinting codes: higher rates, quick accusations," Ph.D. dissertation, Simon Fraser University, Fall 2010.

[7] E. Knill, A. Schliep, and D. C. Torney, "Interpretation of pooling experiments using the Markov chain Monte Carlo method," *J Comput Biol*, vol. 3, no. 3, pp. 395–406, 1996.

[8] D. Ge, J. Idier, and E. L. Carpentier, "Enhanced sampling schemes for MCMC based blind Bernoulli-Gaussian deconvolution," *Signal Processing*, vol. 91, no. 4, pp. 759–772, 2011.

[9] C. Robert and G. Casella, *Monte Carlo statistical methods*. Springer Verlag, 2004.

[10] M. R. Gupta and Y. Chen, "Theory and use of the EM algorithm," *Foundations and Trends in Signal Processing*, vol. 4, no. 3, pp. 223–296, 2010.