BLIND DECODER FOR BINARY PROBABILISTIC TRAITOR TRACING CODES

Luis Pérez-Freire, Teddy Furon

BLIND DECODER FOR BINARY PROBABILISTIC TRAITOR TRACING CODES

Luis Pérez-Freire[*]

GRADIANT
ETSI Telecom., Lagoas Marcosende s/n
Vigo, Spain
lpfreire@gradiant.org

Teddy Furon

Thomson Security Lab
1, av. Belle Fontaine
Cesson Sévigné, France
teddy.furon@thomson.net

ABSTRACT

This paper presents a new decoder for probabilistic binary traitor tracing codes which is based on classical hypothesis testing and estimation theory. This new decoder is blind, in the sense of ignoring a priori the collusion attack it is facing. It basically performs a joint estimation of the collusion channel and the probability that each user takes part in the collusion. The experimental results shown in the paper have been obtained with the classical Tardos code, although the proposed decoder works with arbitrary probabilistic binary codes. Another remarkable advantage of this blind decoder is its ability to successfully cope with collusion channels far more general than the classical Marking Assumption, including channels that produce erasures and random decoding errors.

Index Terms— Traitor tracing, Tardos code, collusion channel estimation, likelihood ratio, optimal decoding.

1. INTRODUCTION

Probabilistic codes, which have been recently introduced in the field of traitor tracing, are beginning to be well understood. Since the seminal work by Tardos [1], a number of works studying their properties and fundamental limits have blossomed out during the last years [2],[3]. However, these works mostly address the issue of coding performance assuming that a "joint" decoder will do the accusation work. Joint decoders, which have been recently proved optimal in an information-theoretic framework [4], [2], [3], [5], perform joint accusations considering groups of colluders. The main problem with joint decoders is that they are too complex to implement in practice. To the best of our knowledge, all traitor tracing decoders available in the literature belong to the class of "simple decoders", i.e. they output decisions for each user independently of the others. This class of decoders, albeit suboptimal, is affordable from the computational point of view.

In this paper we are concerned with "simple" decoding of probabilistic traitor tracing codes, such as that proposed by Tardos [1]. The decoding algorithms proposed by Tardos [1] and Skoric [6] pose the remarkable property of providing stable performance independently of the collusion channel, provided that the latter fulfills the so-called "marking assumption" [7]. This decoding strategy can be considered as "agnostic", since it does not use any information about the collusion channel. We will refer to it in the sequel simply as "Tardos decoding". The main advantages of these decoders are that they are simple to implement and that the bound on the minimal code length for satisfying a required error probability holds whatever the collusion attack. However, this decoding strategy suffers from three major drawbacks:

1. Tardos decoding has a too strong coupling between the accusation algorithm and the code generation, since it is designed to work with an specific time-sharing distribution, which is the one originally proposed by Tardos. However, some recent works have shown, from a game-theoretic viewpoint, that the optimal time-sharing distribution is strongly dependent on the number of colluders. In such conditions, the stable performance property of the agnostic decoder is no longer granted.
2. When the collusion strategy deviates from the marking assumption, performance stability for the agnostic decoder does not hold. For instance, when the channel introduces random errors, decoding performance rapidly decreases, even if the errors occur with very small probability. This is not surprising as the code construction and the agnostic decoders were originally designed for solving the Boneh-Shaw problem [7].
3. The agnostic decoders are suboptimal from a hypothesis testing theory viewpoint, especially as the number of colluders increases, as shown in [8].

Thus, although the code construction is provably good, Tardos decoding is in general highly suboptimal. The approach proposed in this paper is radically different in that it relies essentially on maximum likelihood decoding, although important relaxations are introduced in order to lower the decoding complexity. The decoder presented in this paper is basically a simplified version of the iterative decoder proposed in [8], which was shown to outperform the existing Tardos decoders. Such iterative decoder performs a joint estimation-decoding by means of the well known Expectation-Maximization algorithm [9], where some simplifications are introduced. Even with such simplifications, the complexity of the iterative decoder makes it not suitable to scenarios where a huge number of users is involved. The present decoder introduces further simplifications that lower decoding complexity, at the expense of a presumable performance loss. Although the decoder is applicable to arbitrary probabilistic traitor tracing codes, we focus without loss of generality on the continuous time-sharing distribution proposed by Tardos [1]. The achievable rate of the resulting code has been studied in [10]. As will be seen, using the proposed decoder is possible to work at rates close to the theoretical upper limit, whilst providing small error probabilities.

The rest of the paper is organized as follows: Section 2 provides the required mathematical formulation of the problem. Section 3 presents the rationale of our decoding approach and the details of the proposed decoder. Section 4 is concerned with the experimental evaluation, and finally Section 5 gives some concluding remarks.

[*]The first author performed this work while at Thomson Security Lab.

2. PROBLEM FORMULATION

The basic notation that will be used throughout the paper is as follows. The number of users and the length of the code are denoted by n and m, respectively. Thus, the coding rate is given by $R = \log(n)/m$. The probability that the random variable A defined over the alphabet $\mathcal{A}$ takes the occurrence a is denoted by $\mathbb{P}_A[a]$. The cardinality of the set $\mathcal{A}$ is denoted by $|\mathcal{A}|$. Vectors and matrices are written in boldface.

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 $\mathbf{x}_j = (x_{j1}, \cdots, x_{jm})$ identifying user j is composed of m binary symbols. Indeed, these symbols are drawn independently such that $\mathbb{P}_{X_{ji}}[1] = p_i$, $\forall i = 1, \dots, m$. The values $\{P_i\}_{i=1,\dots,m}$ 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 $\{p_i\}_{i=1,\dots,m}$ of these random variables are drawn once for all at the initialization of the code, and they constitute its secret key which is shared between encoder and decoder.

2.2. The collusion channel

Denote the subset of colluder indices by $\mathcal{C} = \{j_1, \cdots, j_c\}$, and $\mathbf{x}_\mathcal{C} = \{\mathbf{x}_{j_1}, \dots, \mathbf{x}_{j_c}\}$ 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. The symbols y_i belong to an alphabet $\mathcal{Y} = \{0, 1, \dots, |\mathcal{Y}|\}$, which is not necessarily binary.

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). For binary codes, this type is fully defined by the following sufficient statistic: the number Σ_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 (e.g. 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 a parameter $\boldsymbol{\theta}$ which is in turn a concatenation of parameter vectors as follows: $\boldsymbol{\theta} = (\boldsymbol{\theta}[1], \boldsymbol{\theta}[2], \dots, \boldsymbol{\theta}[|\mathcal{Y}|])$, where

$$\boldsymbol{\theta}[y] = (\mathbb{P}_Y[y|\Sigma = 0], \dots, \mathbb{P}_Y[y|\Sigma = c]), \text{ for } y \in \mathcal{Y}. \quad (1)$$

There is thus an infinity of collusion attacks, but all of them are defined by $(c+1)|\mathcal{Y}|$ real values in the hypercube $[0,1]^{(c+1)|\mathcal{Y}|}$. In general, $\boldsymbol{\theta}$ is restricted to some subset $\mathcal{P}^c$ of $[0,1]^{(c+1)|\mathcal{Y}|}$ defined as

$$\begin{aligned} \mathcal{P}^c \quad \triangleq \quad & \{\boldsymbol{\theta} \in [0,1]^{(c+1)|\mathcal{Y}|} \text{ such that } dist(\mathbf{x}_\mathcal{C}, \mathbf{y}) < \delta, \\ & \sum_{y \in \mathcal{Y}} \mathbb{P}_Y[y|\sigma] = 1, \text{ for all } \sigma = 0, \dots, c\}. \end{aligned} \quad (2)$$

where $dist(\mathbf{x}_\mathcal{C}, \mathbf{y})$ is an appropriately defined distortion metric and δ the admissible distortion. Notice also that the number of degrees of freedom in $\boldsymbol{\theta}$ is in general much lower than $(c+1)|\mathcal{Y}|$. For instance, if $\mathcal{Y} = \{0, 1\}$, then $\boldsymbol{\theta}$ has no more than c+1 degrees of freedom because $\mathbb{P}_Y[0|\Sigma = \sigma] + \mathbb{P}_Y[1|\Sigma = \sigma] = 1$.

2.3. Decoding and error measures

Once the sequence $\mathbf{y}$ is extracted from the pirated content, a decoding algorithm assigns to each sequence $\{\mathbf{x}_j\}_{i=1,\dots,m}$ a score. User j is deemed guilty if the score of $\mathbf{x}_j$ is above a certain threshold τ. Let $\hat{\mathcal{C}}$ be the estimated set of colluders. Two error events are defined:

- False alarm: at least one innocent user belongs to $\hat{\mathcal{C}}$.
- False negative: none of the colluders is caught, i.e. $\mathcal{C} \cap \hat{\mathcal{C}} = \emptyset$.

Error probabilities are correspondingly denoted by $P_{FA}(\tau)$, $P_{FN}(\tau)$.

The most popular decoding algorithm is the symmetric version of the Tardos decoding function proposed by B. Skoric et al. [6].

3. THE JOINT ESTIMATOR-DECODER

3.1. Main rationale: the informed decoder

According to classical hypothesis testing theory [11, Chap. 12], the accusation is formulated as a test based on the observations $(\mathbf{x}_j, \mathbf{y})$ that consists of checking which of the two following hypothesis is true:

- $\mathcal{H}_0$: User j is innocent,
- $\mathcal{H}_1$: User j is a colluder.

The Neyman-Pearson theorem tells us that the optimal hypothesis testing is implemented by means of the log-likelihood ratio test:

$$L(\mathbf{y}, \mathbf{x}_j) = \frac{1}{m} \log\left(\frac{\mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_1}[\mathbf{y}, \mathbf{x}_j|\mathbf{p}, \boldsymbol{\theta}]}{\mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_0}[\mathbf{y}, \mathbf{x}_j|\mathbf{p}, \boldsymbol{\theta}]}\right). \quad (3)$$

Under hypothesis $\mathcal{H}_0$, $\mathbf{y}$ and $\mathbf{x}_j$ are statistically independent, so the above expression can be simplified to

$$L(\mathbf{y}, \mathbf{x}_j) = \frac{1}{m} \sum_{i=1}^{m} \log\left(\frac{\mathbb{P}_{Y_i|}[y_i|x_{ji}, p_i, \boldsymbol{\theta}]}{\mathbb{P}_{Y_i}[y_i|p_i, \boldsymbol{\theta}]}\right). \quad (4)$$

Let us define the vector $\mathbf{q}_\Sigma(p_i)$ as

$$\mathbf{q}_\Sigma(p_i) \triangleq (\mathbb{P}_\Sigma[0|p_i], \dots, \mathbb{P}_\Sigma[c|p_i]), \quad (5)$$

with $\mathbb{P}_\Sigma[\sigma|p_i] = \binom{c}{\sigma} p_i^\sigma (1-p_i)^{c-\sigma}$ the probability that c colluders have σ times the symbol '1' at index i, conditioned on p_i. Hence,

$$\mathbb{P}_{Y_i}[y_i|p_i, \boldsymbol{\theta}] = \sum_{\sigma=0}^{c} \mathbb{P}_{Y_i}[y_i|\sigma] \mathbb{P}_{\Sigma_i}[\sigma|p_i] = \boldsymbol{\theta}[y_i] \mathbf{q}_\Sigma^T(p_i), \quad (6)$$

where superindex ^{T} means transposed. The other conditional probability involved in (4) is given by

$$\mathbb{P}_{Y_i}[y_i|x, p_i, \boldsymbol{\theta}] = \sum_{\sigma=x}^{c-1+x} \mathbb{P}_{Y_i}[y_i|\sigma] \binom{c-1}{\sigma-x} p_i^{\sigma-x} (1-p_i)^{c-1+2x-\sigma}. \quad (7)$$

Unfortunately, this optimal decoder is not applicable as such in practice, since it needs to know in advance the collusion channel $\boldsymbol{\theta}$. This problem is tackled below.

3.2. The blind decoder

We proposed in [8] a blind decoder that performs iteratively a joint estimation of the collusion channel and the probability of being guilty for each user. Such decoder, based on the well-known Expectation-Maximization algorithm [9], allows to approximate the performance of the optimal informed decoder. Nonetheless, its computational complexity is prohibitive when applied to scenarios where a huge number of users is involved, since in each iteration we must optimize a function whose evaluation requires $O(nmc)$ elementary operations. Performing an accusation with the blind decoder presented here is comprised of three steps.

3.2.1. Step 1

The first step consists in estimating the collusion channel relying only on the pirated sequence $\mathbf{y}$ and the secret sequence $\mathbf{p}$ shared between encoder and decoder. We take advantage of the fact that information about the collusion strategy leaks from the observed $\{\mathbf{y}, \mathbf{p}\}$. Assuming the collusion was performed by k dishonest users, the first step is formulated as a Maximum Likelihood estimator:

$$\hat{\boldsymbol{\theta}}^k = \arg\max_{\boldsymbol{\theta}\in\mathcal{P}^k} J(\boldsymbol{\theta}). \tag{8}$$

Let us define $\mathcal{L}_j$ as the set of indices where the pirated sequence $\mathbf{y}$ has the symbol $j \in \mathcal{Y}$. $J(\boldsymbol{\theta})$ can be compactly expressed as

$$J(\boldsymbol{\theta}) = \sum_{j\in\mathcal{Y}} \sum_{i\in\mathcal{L}_j} \log(\boldsymbol{\theta}[j]\mathbf{q}_\Sigma^T(p_i)), \tag{9}$$

where $\boldsymbol{\theta}[j]$ and $\mathbf{q}_\Sigma^T(p_i)$ have been defined in (1) and (5), respectively. The functional $J(\boldsymbol{\theta})$ is concave in $\boldsymbol{\theta}$, so the problem is a convex problem, subject to a convex constraint as long as $\mathcal{P}^c$ is convex. Therefore, it can be solved by standard convex optimization methods. In particular, we have chosen a constrained conjugate gradient descent algorithm. Since we ignore a priori the number of colluders, c, we perform the optimization (8) for a range of collusion sizes, thus producing a set of candidate collusion channels $\{\hat{\boldsymbol{\theta}}^1, \ldots, \hat{\boldsymbol{\theta}}^{c_{\max}}\}$.

3.2.2. Step 2

In the second step, the blind decoder performs a joint estimation of the most likely colluders and the most likely collusion channel among the candidates. For a given $\hat{\boldsymbol{\theta}}^k$, the probability that user j is guilty can be computed as

$$\pi_j(\hat{\boldsymbol{\theta}}^k) = \mathbb{P}[\mathcal{H}_1|\mathbf{y}, \mathbf{x}_j, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]$$
$$= \frac{\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_1}[\mathbf{y}|\mathbf{x}_j, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]\mathbb{P}[\mathcal{H}_1]}{\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_1}[\mathbf{y}|\mathbf{x}_j, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]\mathbb{P}[\mathcal{H}_1] + \mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_0}[\mathbf{y}|\mathbf{x}_j, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]\mathbb{P}[\mathcal{H}_0]}. \tag{10}$$

Since we assume there are k colluders out of n users, we have $\mathbb{P}[\mathcal{H}_1] = k/n$ and $\mathbb{P}[\mathcal{H}_0] = 1 - k/n$, and we arrive at

$$\pi_j(\hat{\boldsymbol{\theta}}^k) = \frac{k}{k + (n-k)\dfrac{\mathbb{P}_{\mathbf{Y}}[\mathbf{y}|\mathbf{p}, \hat{\boldsymbol{\theta}}^k]}{\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_1}[\mathbf{y}|\mathbf{x}_j, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]}}. \tag{11}$$

The probabilities appearing in this last equation have already been expressed in Sect. 3.1. Now, define the random variable S_j that takes the values 1 or 0, respectively, when user j is a colluder or an innocent. We compute an approximation to the likelihood of $\hat{\boldsymbol{\theta}}^k$ given the codebook as[1]

$$\begin{aligned} Q(\hat{\boldsymbol{\theta}}^k) &= \log\left(\mathbb{P}_{\mathbf{Y}}[\mathbf{y}|\mathbf{p}, \hat{\boldsymbol{\theta}}^k]\right) \\ &+ \sum_{j=1}^n \pi_j(\hat{\boldsymbol{\theta}}^k)\log\left(\mathbb{P}_{\mathbf{X}_j,S_j}[\mathbf{x}_j, 1|\mathbf{y}, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]\right) \\ &+ \sum_{j=1}^n (1-\pi_j(\hat{\boldsymbol{\theta}}^k))\log\left(\mathbb{P}_{\mathbf{X}_j,S_j}[\mathbf{x}_j, 0|\mathbf{y}, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]\right). \end{aligned} \tag{12}$$

With some more work:

$$\mathbb{P}_{\mathbf{X}_j,S_j}[\mathbf{x}_j, 0|\mathbf{y}, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]$$
$$= (1-\frac{k}{n})\mathbb{P}_{\mathbf{X}_j}[\mathbf{x}_j|\mathbf{p}] = (1-\frac{k}{n})\prod_{i=1}^m p_i^x(1-p_i)^{(1-x)}, \tag{13}$$

$$\mathbb{P}_{\mathbf{X}_j,S_j}[\mathbf{x}_j, 1|\mathbf{y}, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]$$
$$= \frac{k}{n}\cdot\frac{\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_1}[\mathbf{y}|\mathbf{x}_j, \mathbf{p}, \hat{\boldsymbol{\theta}}^k]\mathbb{P}_{\mathbf{X}_j}[\mathbf{x}_j|\mathbf{p}]}{\mathbb{P}_{\mathbf{Y}}[\mathbf{y}|\mathbf{p}, \hat{\boldsymbol{\theta}}^k]} \tag{14}$$

3.2.3. Step 3

We select the index of the most likely collusion channel as

$$\hat{c} = \arg\max_{k=1,\ldots,c_{\max}} Q(\hat{\boldsymbol{\theta}}^k), \tag{15}$$

Finally, we estimate the group of colluders as

$$\hat{\mathcal{C}} = \{j \in \{1, \ldots, n\} \text{ such that } \pi_j(\hat{\boldsymbol{\theta}}^{\hat{c}}) > \tau\}. \tag{16}$$

4. EXPERIMENTAL EVALUATION

We consider several scenarios of practical relevance that illustrate how powerful is the proposed blind decoder. In all cases, its performance is compared to that of the optimal (informed) decoder and the Tardos decoder. For the latter, we have implemented the symmetric version proposed by Skoric et al [6], which has been shown to improve significantly the original proposal by Tardos [1]. The value of the variable $c_{\max}$, introduced in Sect. 3.2.1 is fixed to 10 in all cases.

4.1. Marking Assumption

This is the classical Boneh-Shaw problem [7]. The colluders forge the pirated sequence $\mathbf{y}$ by assembling pieces from their personal copies. Obviously, $\mathcal{Y} = \{0, 1\}$. The following simplifications hold:

1. The parameter vector defining the collusion channel is completely defined by $\boldsymbol{\theta}[1]$. Thus, there are no more than $c+1$ degrees of freedom.
2. The Marking Assumption enforces that $\mathbb{P}_Y[1|\Sigma = 0] = 0$ and $\mathbb{P}_Y[1|\Sigma = c] = 1$. Hence, the number of degrees of freedom is actually $c-1$.[2]

[1]Details will be given in the journal version, currently under preparation.
[2]This restriction has not been implemented in our practical evaluation. Notice that better results could have been obtained in such case.

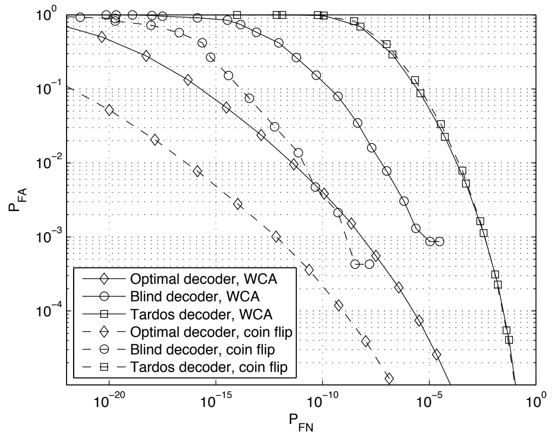

Fig. 1. Decoding performance under the Marking Assumption for $c = 8, n = 10^5, m = 4500$.

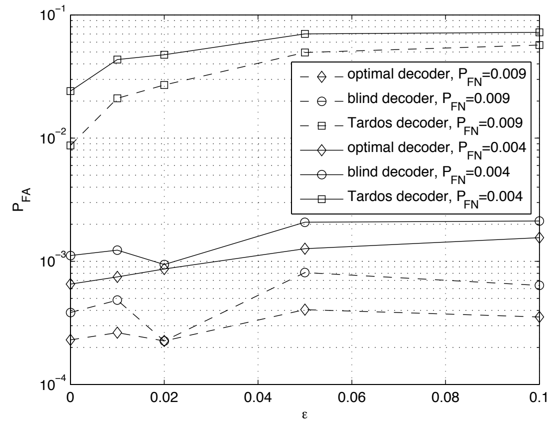

Fig. 2. Probability of false alarm under Marking Assumption followed by a binary symmetric channel with parameter ε. $c = 3$, $n = 10^4$.

Fig. 1 shows the plot of $P_{FA}(\tau)$ vs. $P_{FN}(\tau)$ in different scenarios. In one case, the colluders apply the Worst Case Attack, which is the attack minimizing the achievable rate of the code [10]. In the other case, they simply apply a "coin flip attack" [10], i.e. they decide the output bit upon a coin flip whenever they are allowed by the Marking assumption. Notice in all cases the stability of Tardos decoding, and the big gap with regard to the optimal and blind decoders.

4.2. Marking Assumption + random errors

The pirated sequence $\mathbf{y}$, forged as in Sect.4.1, undergoes a posteriori a binary symmetric channel (BSC) that flips with probability ε the bits in $\mathbf{y}$ independently for each index $i \in \{1, \ldots, m\}$. Yet the output alphabet remains $\mathcal{Y} = \{0, 1\}$. Note that Marking Assumption no longer holds, i.e. $\mathbb{P}_Y[1|\Sigma = 0] \geq 0$ and $\mathbb{P}_Y[1|\Sigma = c] \leq 1$. This model is applicable, for instance, to multimedia scenarios where each bit of the sequences $\mathbf{x}_j$ is embedded in one block of the multimedia content by means of a watermarking technique, and the colluders forge the pirated content by exchanging their blocks and applying some postprocessing like a lossy compression. Fig. 2 shows the value of P_{FA} as a function of ε when P_{FN} is fixed to a given value and there are $c = 3$ colluders applying the Worst Case Attack. The code length in all cases is chosen so that the rate is 50% of the theoretical achievable rate.

4.3. Marking assumption + erasures

The colluders apply now a compound attack. With probability $\mathbb{P}_E[e|\sigma]$, they put a symbol which is neither a '0' nor a'1', so it is treated as an erasure. With probability $1 - \mathbb{P}_E[e|\sigma]$, they apply a majority attack, i.e. they put the most frequent symbol they have. Thus, the output alphabet becomes $\mathcal{Y} = \{0, 1, e\}$. This setup is applicable to traitor tracing scenarios with multimedia content:

1. The fingerprinted copy of user j is generated by dividing the content into nonoverlapping blocks, and embedding in each block one bit of $\mathbf{x}_j$ with a zero-bit watermarking scheme. Two different secret keys are used for embedding 0 or 1 correspondingly.
2. The colluders generate the pirated content by averaging their copies with equal weights, and then they apply a lossy compression. We assume that the bit with the highest energy has a higher probability of being detected[3].
3. The watermark detector recovers the bit embedded in each block of the pirated copy. In some blocks it does not detect any bit due to the lossy compression; in some other blocks it detects both bits. If any of this things happen, then the decoder declares an erasure.

In the experiments, we evaluate an scenario with $c = 7$ colluders. In the parameter vector defining the collusion channel, $\boldsymbol{\theta}[0]$ and $\boldsymbol{\theta}[1]$ model the averaging operation, whereas $\boldsymbol{\theta}[e]$ models the lossy compression, for which we have chosen $\boldsymbol{\theta}[e] = (1 - \varepsilon_e,\ 1 - 0.85\varepsilon_e,\ 1-0.7\varepsilon_e,\ 1-0.6\varepsilon_e,\ 1-0.6\varepsilon_e,\ 1-0.7\varepsilon_e,\ 1-0.85\varepsilon_e,\ 1-\varepsilon_e)$. Parameter $\varepsilon_e \in [0, 1]$ is used to tune the probability of having an erasure, $\mathbb{P}_E[e] = \mathbb{E}_P\{\boldsymbol{\theta}[e]\mathbf{q}_\Sigma^T(p)\}$, where $\mathbb{E}_P\{\}$ denotes expectation in P. $\mathbb{P}_E[e]$ increases as ε_e decreases. For the chosen $\boldsymbol{\theta}[e]$, the probability of having an erasure increases as σ approaches $\sigma/2$; the rationale is that the averaging operation reduces the energy of the bits embedded in each block when they are different (i.e. $0 < \sigma < 1$).

Fig. 3 shows the probability of false alarm as a function of $\mathbb{P}_E[e]$ when P_{FN} is fixed to a 0.02 and 0.005. The coding rate lies between 30 and 50% of the achievable rate for the considered collusion channels. As can be seen, the performance of the blind decoder degrades smoothly as the probability of erasure is increased, and its error probability is always remarkably lower than that of the Tardos decoder, for which $P_{FA} \approx 1$ in all cases. Notice that the Tardos decoder simply disregards the indices containing an erasure.

4.4. Marking assumption + erasures + double symbols

The experimental setup is the same as in Sect. 4.3, but when the watermark detector detects more than one bit in the ith block of the content, its output y_i is then a "double symbol" d. Hence, the output alphabet is comprised now of four symbols: $\mathcal{Y} = \{0, 1, e, d\}$.

We evaluate an scenario with $c = 7$ colluders, with $\boldsymbol{\theta}[d] = (0,\ 0.27\times(1-0.85\varepsilon_s),\ 0.53\times(1-0.7\varepsilon_s),\ 0.8\times(1-0.6\varepsilon_s),\ 0.8\times(1-0.6\varepsilon_s),\ 0.53\times(1-0.7\varepsilon_s),\ 0.27\times(1-0.85\varepsilon_s),\ 0)$. Finally, $\boldsymbol{\theta}[e]$ and ε_s are fixed so that the probability of detecting an erasure

[3]This would be the case if the bits were embedded with an spread spectrum watermarking scheme

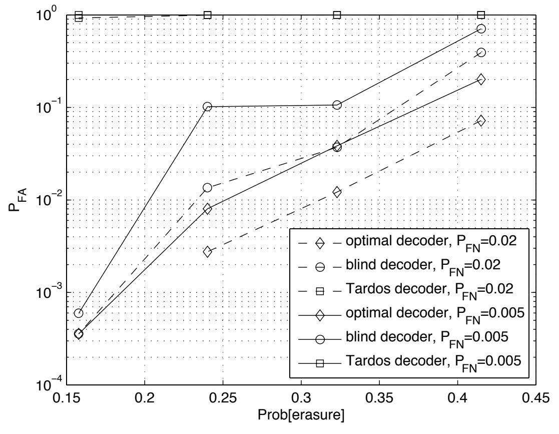

Fig. 3. Decoding performance under Marking Assumption + erasures, for $c = 7$, $m = 900$, and $n = 10^4$ users.

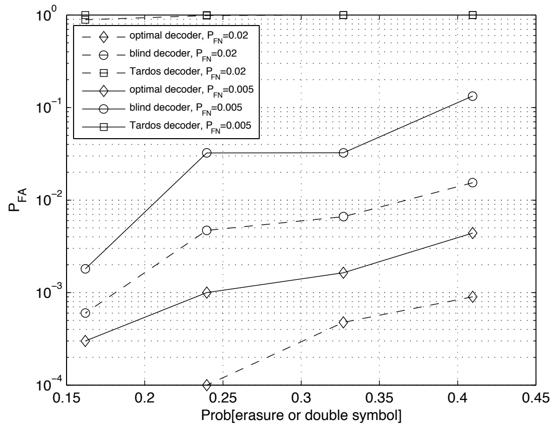

Fig. 4. Decoding performance under Marking Assumption + erasures + double detections, for $c = 7$, $m = 900$, and $n = 10^4$ users.

or a double symbol given σ is the same as $\mathbb{P}_E[e|\sigma]$ in Sect. 4.3. Hence, some erasures obtained in Sect. 4.3 remain as erasures but others are transformed into double symbols. Results are shown in Fig. 4. A comparison between the latter and Fig. 3 reveals that the blind decoder is able to successfully take advantage of this additional information, as P_{FA} is in general significantly lower in Fig. 4 than in Fig. 3. However, for small probability of erasure or double symbol (left part of the curves), performance is similar or slightly worse. The reason could be that the number of samples with erasures and double symbols in such case is too small so as to estimate the collusion channel with sufficient accuracy. As for the Tardos decoder, since it disregards indices with both erasures and double symbols, it remains stuck at $P_{FA} \approx 1$ as in Fig. 3. The coding rate lies between 30% and 40% of the achievable rate for the considered collusion channels.

5. CONCLUDING REMARKS

The blind decoder for probabilistic traitor tracing codes presented in this paper has shown promising results through its application to a classical Tardos code. Although the gap with regard to the optimal decoder is significant in some cases, its performance is (at the cost of some additional but affordable computational complexity) remarkably better than that of the existing Tardos decoder, which so far is the main reference in the field. Besides its good performance, the blind decoder poses other advantages such as the possibility of working with arbitrary probabilistic binary codes (not only the Tardos code), and the ability of successfully managing random errors, erasures and double symbols. Hence, it encompasses many more traitor tracing scenarios than the classical Marking Assumption for which the classical Tardos decoder was designed. The preliminary study presented here opens however some important questions, which will be addressed in our future research:

1. The accuracy in estimating the collusion channel, and its impact in decoding performance.
2. How to fix the decoding threshold τ so as to guarantee an objective error probability.
3. The impact of time varying collusion attacks.
4. The possibility of using the proposed decoder as a building block for implementing an affordable approximation of the joint decoder, which is currently a kind of holy graal for optimal traitor tracing [2],[3].

6. REFERENCES

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

[2] Pierre Moulin, "Universal fingerprinting: capacity and random-coding exponents," *IEEE Transactions on Information Theory*, January 2008, Submitted. Preprint available at `http://arxiv.org/abs/0801.3837`.

[3] E. Amiri and G. Tardos, "High rate fingerprinting codes and the fingerprinting capacity," in *Proc. of 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, jan 2009.

[4] N.P. Anthapadmanabhan, A. Barg, and I. Dumer, "On the fingerprinting capacity under the marking assumption," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2678–2689, June 2008.

[5] Y.-W. Huang and P. Moulin, "Saddle-point solution of the fingerprinting capacity game under the marking assumption," in *accepted to ISIT 2009*, jun 2009.

[6] B. Skoric, T. Vladimirova, M. Celik, and J. Talstra, "Tardos fingerprinting is better than we thought," arXiv:cs/0607131v1, July 2006.

[7] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1897–1905, September 1998.

[8] Teddy Furon and Luis Pérez-Freire, "EM decoding of Tardos traitor tracing codes," in *ACM Multimedia Security, MM-SEC'09*, Princeton, NJ, USA, 2009, Submitted.

[9] Todd K. Moon, "The Expectation-Maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, November 1996.

[10] Teddy Furon and Luis Pérez-Freire, "Worst case attacks against binary probabilistic traitor tracing codes," *IEEE Transactions on Information Forensics and Security*, 2009, Submitted, available at `http://arxiv.org/abs/0903.3480`.

[11] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, Wiley series in Telecommunications, 1991.