# EM Decoding of Tardos Traitor Tracing Codes

Teddy Furon, Luis Pérez-Freire

# EM Decoding of Tardos Traitor Tracing Codes

Teddy Furon
Thomson Security Competence Center
1, av. Belle Fontaine
Cesson Sévigné, France
teddy.furon@thomson.net

Luis Pérez-Freire*
Gradiant, ETSI Telecom.
Lagoas Marcosende s/n,
Vigo, Spain
lpfreire@gradiant.org

## ABSTRACT

This paper proposes a major shift in the decoding of probabilistic Tardos traitor tracing code. The goal of the decoder is to accuse colluders but it ignores how they have been mixing their copies in order to forge the pirated content. As originally proposed by Tardos, so far proposed decoders are agnostic and their performances are stable with respect to this unknown collusion attack. However, this stability automatically leads to non-optimality from a detection theory perspective. This is the reason why this paper proposes to estimate the collusion attack in order to approximate the optimal matched decoder. This is done iteratively thanks to the application of the well-known Expectation-Maximization algorithm. We have dropped the stability: the power of our decoding algorithm deeply depends on the collusion attack. Some attacks are worse than others. However, even for the worst collusion channel, our decoder performs better than the original Tardos decoding.

## Categories and Subject Descriptors

D.4.6 [**Software**]: Security and Protection—*Information flow controls*

## General Terms

Algorithms

## Keywords

Traitor tracing, fingerprinting, collusion, iterative decoding, Expectation-Maximization

## 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 [12] becomes more and more popular. The reader will find a pedagogical review of this code in [6]. This code is a probabilistic weak fingerprinting code, where the probability of accusing an innocent is not null but very low. The decoding of this 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 $\epsilon_1$ of accusing an innocent and the probability of missing all colluders $\epsilon_2$. Most of the articles dealing with the Tardos code aim at finding a tight lower bound of the required length of the code for a given $\epsilon_1$ [11, 2]. Other works propose more practical implementations of the Tardos code such as [10]. A recent trend is to analyze this code from the information theory viewpoint, and to tune the parameter (namely the time sharing distribution $f(p)$) at the code generation side to maximize the achievable rate, ie. to achieve capacity [1, 9].

In this paper, we study the other end of the chain: the accusation side. We are concerned with decoding of, in general, probabilistic traitor tracing codes, and in particular, those originally proposed by G. Tardos. The decoding algorithms proposed by Tardos and Skoric [12, 11] pose the remarkable property of providing stable performance independently of the collusion channel, provided that the latter fulfills the so-called "marking assumption" [3]. We called this rationale the "agnostic" decoding strategy since it does not use any information about the collusion channel. Its main advantage is that the bound on the minimal code length holds whatever the collusion attack. However, this decoding strategy suffers from three drawbacks.

First, this agnostic decoding strategy has a too strong coupling between the accusation algorithm and the code generation, since it was designed for an specific $f(p)$. However, some recent works [1] have shown, from a game-theoretic viewpoint, that the optimal time sharing distribution $f(p)$

*The second author contributes to this work while he was at Thomson Security Lab


*MM&Sec'09,* September 7–8, 2009, Princeton, New Jersey, USA.

is strongly dependent on the number of colluders and may significantly deviate from the $f(p)$ originally proposed by Tardos. In such conditions, the stable performance property of the agnostic decoder is no longer granted. Second, the stable performance property doesn't hold either whenever the marking assumption is not enforced. For instance, when the channel introduces random errors, decoding performance rapidly decreases. Third, this agnostic decoding is highly suboptimal from a hypothesis testing theory viewpoint.

Thus, although the code construction is provably good, Tardos decoding provides stable performances independently of the collusion channel at the cost of high suboptimality. This is the reason why we propose a complete shift in the decoding strategy. Instead of ignoring the collusion channel, our decoding estimates it in order to use the matched optimal decoder. This is done iteratively thanks to an application of the well-known Expectation-Maximization algorithm [8]. We have dropped the stable performance property: the power of our decoding algorithm deeply depends on the collusion attack. Some attacks are worse than others. However, we are able to show that even for the worst collusion channel, our decoder performs better than the original Tardos decoding.

## 2. THE MATHEMATICAL MODEL

The probability that the random variable $A$ defined over the alphabet $\mathcal{A}$ takes the occurence $a$ is denoted by $\mathbb{P}_A[a]$. 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 $\mathbb{P}_{X_{ji}}[1] = p_i$, $\forall i \in [m]$, 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 The collusion channel

Denote the subset of colluder indices by $\mathcal{C} = \{j_1, \cdots, j_c\}$, and $\mathbf{X}_\mathcal{C} = \{\mathbf{X}_{j_1}, \ldots, \mathbf{X}_{j_c}\}$ 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 [3]. 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.

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 $\theta_\sigma = \mathbb{P}_Y[1|\Sigma = \sigma]$ which reads as the probability that colluders put a '1' when they have $\sigma$ '1' among their symbols. 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 in the hypercube $[0, 1]^{c-1}$. Here are some examples where $\boldsymbol{\theta}$ is given for $c = 4$:

- Random. The colluders randomly draw one of their symbols: $\boldsymbol{\theta} = (0, 1/4, 1/2, 3/4, 1)$.
- Majority. The colluders put the most frequent symbol: $\boldsymbol{\theta} = (0, 0, 1/2, 1, 1)$.
- Minority. The colluders put the less frequent symbol: $\boldsymbol{\theta} = (0, 1, 1/2, 0, 1)$.
- Coin flip. The colluders flip a coin to decide: $\boldsymbol{\theta} = (0, 1/2, 1/2, 1/2, 1/2, 1)$.
- Worst Case Attack (WCA) is the attack minimizing $\mathbb{E}_P[D(P, \boldsymbol{\theta})]$ and hence the loss (14) (See Sect. 3.1). More details are given in [7]. A minimization algorithm gives $\boldsymbol{\theta} = (0, 0.488, 0.50, 0.512, 1)$.

### 2.3 The agnostic decoder

The most well-spread decoding algorithm is the symmetric version of the Tardos decoding function proposed by B. Skoric et al [11]. Once the sequence $\mathbf{Y}$ is extracted from the pirated content, the algorithm calculates a score for any user: $s_j = \sum_{i=1}^{m} U(y_i, x_{ji}, p_i)$, $j \in [n]$, with

$$U(1,1,p) = \sqrt{\frac{1-p}{p}} \qquad U(0,0,p) = \sqrt{\frac{p}{1-p}} \quad (1)$$

$$U(1,0,p) = -\sqrt{\frac{1-p}{p}} \qquad U(0,1,p) = -\sqrt{\frac{p}{1-p}} \quad (2)$$

Statistically, the scores of the colluders are bigger than the scores of the innocents. The decoding accuses the user with the biggest score, or, as originally proposed by Tardos, the users whose scores are above a given threshold.

Asymptotically, as $m \to \infty$, the scores are distributed according to two Gaussian distribution (basic application of the Central Limit Theorem):

$$\text{Innocent:} \quad S_j \sim \mathcal{N}(0, m), \tag{3}$$

$$\text{Colluder:} \quad S_j \sim \mathcal{N}(2\pi^{-1}mc^{-1}, m(1-4\pi^{-2}c^{-2})). \tag{4}$$

# 3. THE INFORMED DECODER

This section provides some arguments why the agnostic decoding strategy is not optimal.

## 3.1 Kullback Leibler distance

Our first argument is taken from classical hypothesis testing theory [5, Chap. 12]. The accusation is indeed 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 accusation process is composed of two building blocks: the calculus of the score and the comparison to a threshold. By Stein's Lemma, the best asymptotic false positive error exponent is given by the Kullback Leibler distance between the pdfs under hypothesis $\mathcal{H}_1$ and $\mathcal{H}_0$, which is given by

$$D(\mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_1} || \mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_0}). \tag{5}$$

Unless the statistic used in the decision is a "sufficient statistic", the actual asymptotic error exponent of a given test is necessarily smaller.

In the Tardos decoder, it is not possible to evaluate the Kullback Leibler distance of the scores $S_j$ of Sect. 2.3, except in the asymptotical regime when the pdf are deemed Gaussian. Then the Kullback Leibler distance between the pdf of the scores $D(\mathbb{P}_{S_j|\mathcal{H}_1} || \mathbb{P}_{S_j|\mathcal{H}_0})$ is given by:

$$D(\mathbb{P}_{S_j|\mathcal{H}_1} || \mathbb{P}_{S_j|\mathcal{H}_0}) = \frac{2m}{c^2\pi^2} - \left( \log(1 - \frac{4}{\pi^2 c^2}) + \frac{4}{\pi^2 c^2} \right). \tag{6}$$

As this holds for $m \to \infty$, the dominating term is $2mc^{-2}\pi^{-2}$.

The mathematical model of the collusion of Sec. 2.2 allows us to evaluate (5) before the scoring function. First, the memoryless property of the collusion channel and the independence of the bits of the code sequence simplifies not only the distributions under both hypothesis $\mathcal{H}_\ell$ with $\ell \in \{0, 1\}$:

$$\mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_\ell} = \prod_{i=1}^{m} \mathbb{P}_{Y_i,X_{ji}|\mathcal{H}_\ell}, \tag{7}$$

but also the distance:

$$D(\mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_1} || \mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_0}) = \sum_{i=1}^{m} D(\mathbb{P}_{Y_i,X_{ji}|\mathcal{H}_1} || \mathbb{P}_{Y_i,X_{ji}|\mathcal{H}_0}) \tag{8}$$

Under $\mathcal{H}_0$, $X_{ji}$ is statistically independent of $Y_i$:

$$\mathbb{P}_{Y_i,X_{ji}|\mathcal{H}_0}[y, x] = \mathbb{P}_{Y_i}[y]\mathbb{P}_{X_{ji}}[x], \tag{9}$$

with $\mathbb{P}_{X_{ji}}[x] = p_i^x(1-p_i)^{(1-x)}$. The probability of the pirated symbol is slightly more complex and involves the collusion model introduced in Sect. 2.2:

$$\mathbb{P}_{Y_i}[1] = \sum_{\sigma=0}^{c} \theta_\sigma \mathbb{P}_{\Sigma_i}[\sigma], \tag{10}$$

with $\mathbb{P}_{\Sigma_i}[\sigma] = \binom{c}{\sigma} p_i^\sigma (1-p_i)^{c-\sigma}$, the probability that $c$ colluders have $\sigma$ times the symbols '1' at index $i$. Obviously, $\mathbb{P}_{Y_i}[0] = 1 - \mathbb{P}_{Y_i}[1]$.

Under $\mathcal{H}_1$, $X_{ji}$ was used to create $Y_i$:

$$\mathbb{P}_{Y_i,X_{ji}|\mathcal{H}_1}[y, x] = \mathbb{P}_{Y_i|X_{ji},\mathcal{H}_1}[y|x]\mathbb{P}_{X_{ji}}[x]. \tag{11}$$

This conditional probability is slightly different than (10):

$$\mathbb{P}_{Y_i|X_{ji},\mathcal{H}_1}[1|x] = \sum_{\sigma=x}^{c-1+x} \theta_\sigma \binom{c-1}{\sigma-x} p_i^{\sigma-x}(1-p_i)^{c-1+2x-\sigma}. \tag{12}$$

Having the expressions of all the probabilities, it is straight forward to calculate each summand of (8) as a function $D(p_i, \boldsymbol{\theta})$ of $p_i$. Averaging over the ensemble of auxiliary sequences, we obtain:

$$D(\mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_1} || \mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_0}) = m\mathbb{E}_P\left[D(P, \boldsymbol{\theta})\right], \tag{13}$$

with $\mathbb{E}_P\left[a(P)\right]$ the expectation of $a(P)$, ie. $\int_0^1 a(p)f(p)dp$.

The Kullback-Leibler distance of the input sequences is strongly dependent on the collusion channel. The loss due to the use of the agnostic decoder is therefore asymptotically equal to:

$$\lambda(\boldsymbol{\theta}) = m(\mathbb{E}_P\left[D(P, \boldsymbol{\theta})\right] - 2c^{-2}\pi^{-2}). \tag{14}$$

Table 1 shows the loss for different collusion channels which are commonly addressed in the literature and defined in Sec. 2.2. In all cases, the loss represents at least 45% of the achievable rate.

| $c$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| Random | 61 | 66 | 71 | 75 | 78 | 81 | 84 |
| Majority | 61 | 149 | 150 | 217 | 219 | 275 | 276 |
| Minority | 61 | 71 | 145 | 334 | 452 | 668 | 780 |
| Coin flip | 61 | 49 | 51 | 60 | 71 | 82 | 94 |
| WCA | 61 | 45 | 51 | 47 | 52 | 52 | 54 |

**Table 1: Loss $\lambda(\boldsymbol{\theta})$ in percentage for some collusion channels. WCA stands for Worst Case Attack, ie. $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_P\left[D(P, \boldsymbol{\theta})\right]$**

.

## 3.2 The expression of the informed decoder

The Neyman-Pearson theorem tells us that a score based on any monotonically increasing function of the likelihood is optimal in the sense that its loss is null. The likelihood $L(\mathbf{y}, \mathbf{x}_j)$ is the ratio

$$L(\mathbf{y}, \mathbf{x}_j) = \frac{\mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_1}[\mathbf{y}, \mathbf{x}_j]}{\mathbb{P}_{\mathbf{Y},\mathbf{X}_j|\mathcal{H}_0}[\mathbf{y}, \mathbf{x}_j]} = \prod_{i=1}^{m} \frac{\mathbb{P}_{Y_i|X_{ji},\mathcal{H}_1}[y_i|x_{ji}]}{\mathbb{P}_{Y_i}[y_i]}, \tag{15}$$

which can be evaluated thanks to (10) and (12). Thus, the optimal decoder computes the likelihood ratio for each user sequence, and deems that those whose likelihood is above a certain threshold are colluders.

Nevertheless, this optimal decoder is a priori not realizable since it needs to know in advance the collusion size $c$ and the collusion channel model $\boldsymbol{\theta}$. We named it the *informed* decoder. Moreover, the pdf of this likelihood ratio is certainly dependent of the collusion channel, so the accusation threshold cannot be a priori fixed to guarantee a given probability of error. Instead, this threshold must be

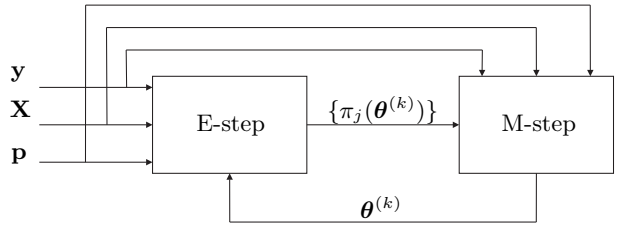



**Figure 2: Block diagram of the iterative decoder.**

a function of $\boldsymbol{\theta}$. Once again, since the collusion channel is not known, this suggests that the informed decoder cannot be implemented.

Figure 1 shows the ROC (Probability of false alarm vs. probability of false negative) for the agnostic decoder and the informed decoder, illustrating the huge gap. The error probabilities have been computed using the rare event probability estimator proposed in [7].

# 4. THE E.-M. DECODER

This section shows a possible implementation of a decoder that tries to approximate the optimal decoder by means of estimating the collusion channel. We propose to apply the Expectation-Maximization algorithm [8] to our problem. The only prior work we are aware of is [4] where the authors propose a decoding algorithm *a la* E.-M. or in the spirit of the E.-M. However, they do not use any likelihood functions but adaptive scoring Tardos functions, which brings a lot of sub-optimality.

As explained in Sect. 3, an approach based on collusion channel estimation opens the door to powerful decoding. The key idea of the EM decoder is to perform an iterative learn and match strategy:

1. Given an estimated collusion channel, use its matched decoder to suspect some colluders,

2. Given some suspected colluders, their sequences and the pirated sequence, build a more accurate estimate of the collusion channel.

In other words, as we better estimate the collusion channel, we better accuse dishonest users (i.e. we accuse less innocents, and miss less colluders), and in turn, we better estimate what they have been doing as a collusion process.

Another argument is that the decoder gets a mixture of observations $\{\mathbf{x}_j\}$ which belong to two families 'innocent' or 'colluder'. Therefore, decoding boils down to estimating the hidden state, a.k.a. 'latent variable' in E.-M. literature, of each observation. This is indeed the event $\mathcal{H}_1$ that user $j$ is a colluder. Our decoding problem is thus very similar to a mixture modelling which is a typical application of the E.-M. algorithm.

The main blocks and variables of the iterative decoder are shown in Figure 2. A detailed explanation is given below.

## 4.1 E-step

The goal of the E-step is to have a better guess about the identities of the colluders. At iteration $k+1$, its inputs are the sequences of the users and the pirated sequence. It also benefits from the estimate $c^{(k)}$ of the collusion size and an estimate $\boldsymbol{\theta}^{(k)}$ of the collusion channel. It simply calculates the probability $\pi_j(\boldsymbol{\theta}^{(k)})$ that user $j$ is guilty according to this collusion channel:

$$\begin{aligned}\pi_j(\boldsymbol{\theta}^{(k)}) &= \mathbb{P}[\mathcal{H}_1|\mathbf{y},\mathbf{x}_j] \\ &= \frac{\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_1}[\mathbf{y}|\mathbf{x}_j]\mathbb{P}[\mathcal{H}_1]}{\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_1}[\mathbf{y}|\mathbf{x}_j]\mathbb{P}[\mathcal{H}_1]+\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_0}[\mathbf{y}|\mathbf{x}_j]\mathbb{P}[\mathcal{H}_0]},\end{aligned} \tag{16}$$

where the application of the Bayes rules lead to the second line. Since there are $c^{(k)}$ colluders out of $n$ users, we have $\mathbb{P}[\mathcal{H}_1] = c^{(k)}n^{-1}$ and $\mathbb{P}[\mathcal{H}_0] = 1-c^{(k)}n^{-1}$. Note that we can integrate in these prior probabilities suspicion for some user based on geographical or behavioral arguments. Under $\mathcal{H}_0$, $\mathbf{Y}$ is independent of $\mathbf{X}_j$, and $\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_0}[\mathbf{y}|\mathbf{x}_j] = \mathbb{P}_{\mathbf{Y}}[\mathbf{y}]$.

$$\pi_j(\boldsymbol{\theta}^{(k)}) = \frac{c^{(k)}}{c^{(k)}+(n-c^{(k)})\frac{\mathbb{P}_{\mathbf{Y}}[\mathbf{y}]}{\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_1}[\mathbf{y}|\mathbf{x}_j]}}, \tag{17}$$

The probabilities appearing in this last equation have already been expressed in the previous section (just replace $\boldsymbol{\theta}$ by $\boldsymbol{\theta}^{(k)}$).

## 4.2 M-step

The goal of the M-step is to have a better guess about the collusion process. At iteration $k+1$, its inputs are the sequences of the users, the pirated sequences, the previous estimation of the collusion channel $\boldsymbol{\theta}^{(k)}$ and the estimation of the identities of the colluders thanks to probabilities $\{\pi_j(\boldsymbol{\theta}^{(k)})\}_{j=1}^n$ evaluated in the E-step.

According to the classical E.-M. formulation, we can refine the estimate of the collusion channel by maximizing in $\boldsymbol{\theta}$ the following functional

$$Q(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}) = \sum_{\mathbf{s}} \mathbb{P}_{\mathbf{S}}[\mathbf{s}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}^{(k)}] \log\left(\mathbb{P}[\mathbf{X},\mathbf{y},\mathbf{s}|\boldsymbol{\theta}]\right). \tag{18}$$

The sequence $\mathbf{s}$ represent the hidden state of the system. It is composed of $n$ binary variables indicating which users are colluders. (18) requires that we consider the $2^n$ possible hidden states, which is of course out of question for a large number of users. We need to introduce the following simplifications:

$$\mathbb{P}[\mathbf{x}_j,\mathbf{y},\mathbf{s}|\boldsymbol{\theta}] = \mathbb{P}[\mathbf{x}_j,\mathbf{y},s_j|\boldsymbol{\theta}], \tag{19}$$

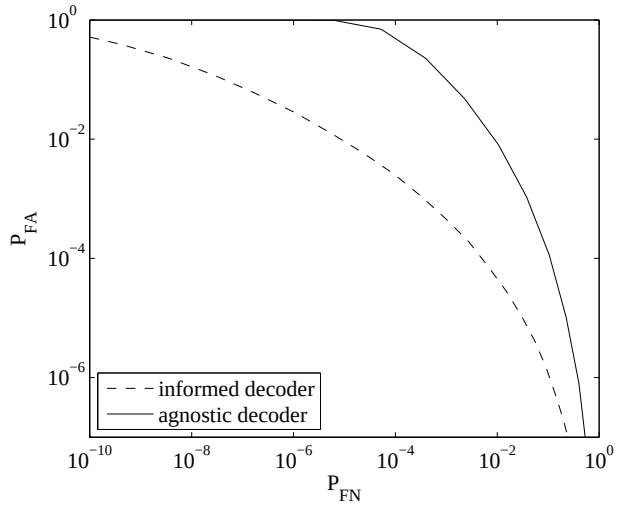
$$\mathbb{P}_{\mathbf{S}}[\mathbf{s}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}] = \prod_{j=1}^{n}\mathbb{P}_{S_j}[s_j|\mathbf{y},\mathbf{x}_j,\boldsymbol{\theta}]. \tag{20}$$

Obviously those equations are not true because guilty users depend on each other, but this is a relaxation needed for having affordable complexity.

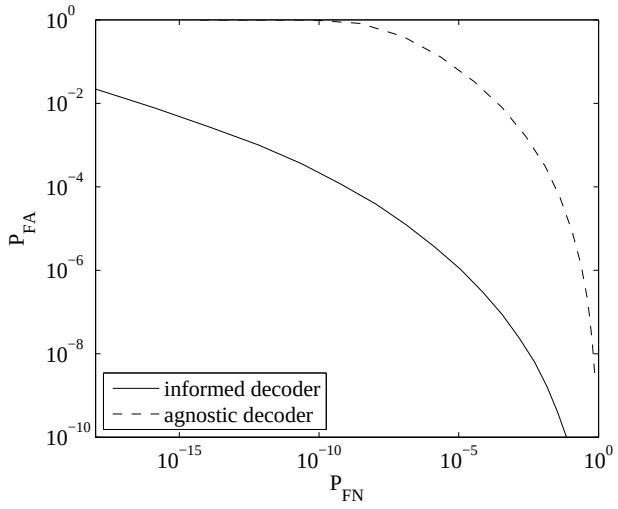
Assume for the moment that $c$ is known, the E-step refines the estimation of the collusion channel with the following simplified functional

$$\boldsymbol{\theta}^{(k+1)} = \arg\max_{\boldsymbol{\theta}} \tilde{Q}_c(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}), \tag{21}$$

with




(a) $c = 3$, $m = 500$, $n = 5000$




(b) $c = 8$, $m = 4500$, $n = 10^5$

**Figure 1: Probabilities of false alarm and false negative with the informed decoder and agnostic decoder, for $c = 3$ (a) and $c = 8$ (b).**

$$\begin{aligned}
&\tilde{Q}_c(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}) \\
&= \sum_{j=1}^{n} \sum_{s_j=\{0,1\}} \mathbb{P}_{S_j}[s_j|\mathbf{y},\mathbf{x}_j,\boldsymbol{\theta}^{(k)}] \log\left(\mathbb{P}_{\mathbf{X}_j}[\mathbf{x}_j,\mathbf{y},s_j|\boldsymbol{\theta}]\right) \\
&= \log\left(\mathbb{P}_{\mathbf{Y}}[\mathbf{y}|\boldsymbol{\theta}]\right) \\
&+ \sum_{j=1}^{n} \pi_j(\boldsymbol{\theta}^{(k)}) \log\left(\mathbb{P}_{\mathbf{X}_j,S_j}[\mathbf{x}_j,1|\mathbf{y},\boldsymbol{\theta}]\right) \\
&+ \sum_{j=1}^{n} (1-\pi_j(\boldsymbol{\theta}^{(k)})) \log\left(\mathbb{P}_{\mathbf{X}_j,S_j}[\mathbf{x}_j,0|\mathbf{y},\boldsymbol{\theta}]\right). \qquad (22)
\end{aligned}$$

With some more work:

$$\begin{aligned}
\mathbb{P}_{\mathbf{X}_j,S_j}[\mathbf{x}_j,0|\mathbf{y},\boldsymbol{\theta}] &= \mathbb{P}_{\mathbf{X}_j}[\mathbf{x}_j](1-c/n), \qquad (23)\\
\mathbb{P}_{\mathbf{X}_j,S_j}[\mathbf{x}_j,1|\mathbf{y},\boldsymbol{\theta}] &= \mathbb{P}_{\mathbf{X}_j}[\mathbf{x}_j|\mathcal{H}_1,\boldsymbol{\theta},\mathbf{y}]c/n \qquad (24)\\
&= \frac{\mathbb{P}_{\mathbf{Y}|\mathbf{X}_j,\mathcal{H}_1}[\mathbf{y}|\mathbf{x}_j]\mathbb{P}_{\mathbf{X}_j}[\mathbf{x}_j]c}{\mathbb{P}_{\mathbf{Y}}[\mathbf{y}]n}. \qquad (25)
\end{aligned}$$

Unfortunately there is no closed form expression for the solution of the M-step, so it must be sought by numerical means. A typical optimization runs as follows:

1. For a parameter $c$ starting from 1 to $c_{\max}$, the function $\tilde{Q}_c(\boldsymbol{\theta}|\boldsymbol{\theta}^{(k)})$ is maximized. Standard optimization algorithms like Newton-Raphson can be used to find out the optimal $\boldsymbol{\theta}^{\star}_c$.
2. Then, the $c_{\max}$ local maxima $\tilde{Q}_c(\boldsymbol{\theta}^{\star}_c|\boldsymbol{\theta}^{(k)})$ are compared in order to isolate the global maximum, and the parameter $\boldsymbol{\theta}^{(k+1)}$ is updated accordingly:

$$\begin{aligned}
c^{(k+1)} &= \arg\max_{c\in[c_{\max}]} \tilde{Q}_c(\boldsymbol{\theta}^{\star}_c|\boldsymbol{\theta}^{(k)}), \qquad (26)\\
\boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{\star}_{c^{(k+1)}} \qquad (27)
\end{aligned}$$

### 4.3 Initialization and termination of the E.-M. algorithm

At the beginning, the accusation process has no idea about the values of the hidden states $\mathbf{s}$, the number of colluders and the collusion attack. We propose to pretend that $c^{(0)} = 2$ and $\theta^{(0)} = [0, 1/2, 1]$, then next comes the E.-step.

The E and M steps are iterated until some termination criterion, usually when $Q(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)})$ is no longer improving or when a maximum number of iteration $k_{\max}$ has been reached. Let $k_f$ be the last iteration, the final decision of the decoder is made by computing the likelihood ratio given in (15) with $\boldsymbol{\theta} = \boldsymbol{\theta}^{(k_f)}$. Notice that the set of probabilities $\{\pi_j(\boldsymbol{\theta}^{(k_f)})\}_{j\in[n]}$ suffice for performing the decision, because (17) is just a monotonic mapping of (15) from $[0,\infty]$ to the interval $[0,1]$. Hence, a threshold $\eta \in [0,\infty]$ in (15) is equivalent to a threshold $T = \frac{c}{c+(n-c)/\eta}$ in (17). We first select the scores bigger than $T$. If the number of these scores is bigger than the estimated collusion size, we only keep the $c^{(k_f)}$ biggest ones. This second rule avoids false alarm (to accuse innocent users) and is typically impossible with a classical Tardos decoding.

## 5. EXPERIMENTAL RESULTS

The experimental work is divided in two parts. First, we investigate whether the proposed iterative decoder correctly estimates the collusion attack. If this is the case, then it would perform as well as the optimal matched decoder. In any case, it doesn't mean that no error would be committed, but that as few errors as theoretically possible would be done. On the contrary, a mismatch in the estimation of the collusion channel doesn't imply a bad decoding as the second part assessing the decoding performances shows.

The experimental setup is the following. There are $n = 1000$ users, we consider code sequences of length $m = 250$ bits, and the number of colluders ranges from 3 to 9. The iterative decoder is assuming that the maximum number of colluders is 9. Hence, the parameter $c_{\max}$ introduced in Sect. 4.2 is set to 9. The maximum number of iterations for the E.-M. decoder is set to $k_{\max} = 5$. Two collusion attacks are considered: Minority vote and the Worst Case Attack. As explained before, the performance of the optimal decoder is largely dependent on the collusion channel. The purpose

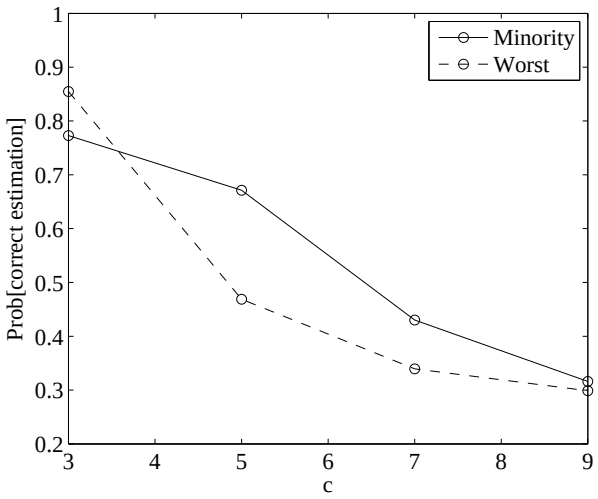



**Figure 3: Probabilities $\mathbb{P}[C^{(k_f)} = c]$ for $m = 400$.**

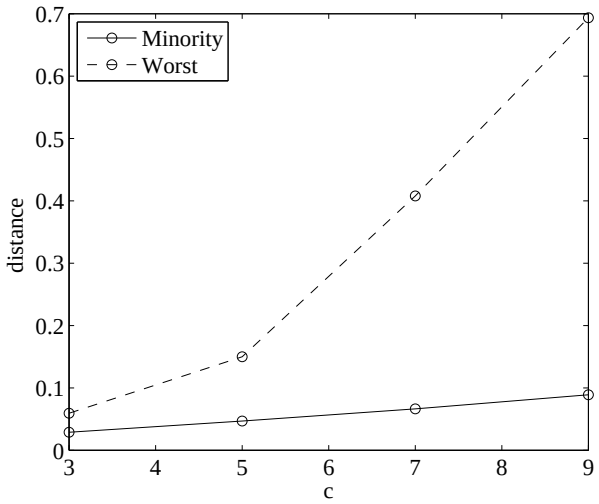



**Figure 4: Distances $d(\boldsymbol{\theta}^{(k_f)}, \boldsymbol{\theta})$ for $m = 400$.**

of focusing in these two particular attacks is to illustrate the performance of our decoder against a "gentle" attack (Minority) and against the worst attack, respectively (c.f. Tab. 1).

## 5.1 Accuracy of the estimated collusion attack

### 5.1.1 Estimation of the collusion size

The first experiment is a Monte-Carlo simulation with $N = 1000$ runs. Each run generates a secret sequence $\mathbf{p}$ and a code $\mathbf{X}$ of $n$ sequences, $c$ of these collude, and the E.-M. decoding proceeds with the received pirated sequence. We measure the probability of correctly estimating the collusion size by the ratio of the number of times where $c^{(k_f)}$ equals $c$ divided by $N$. Fig 3 shows that this probability decreases as the collusion attack is worse: more colluders and/or more harmful process.

### 5.1.2 Estimation of the collusion process

With the same experiment, when the estimate of the collusion side is correct, we measure the accuracy of the estimated collusion attack by the following distance:

$$d(\boldsymbol{\theta}^{(k_f)}, \boldsymbol{\theta}) = \max_{\sigma \in [0,c]} |\theta_\sigma^{(k_f)} - \theta_\sigma| \leq 1 \tag{28}$$

If $d(\boldsymbol{\theta}^{(k_f)}, \boldsymbol{\theta}) = 0$, then the E.-M. succeeds in retrieving the exact value of the collusion attack, and therefore the decoding is optimal. Fig. 4 shows that the estimation becomes inaccurate as $c$ increases. However, there is a big gap between the two attacks: the minority collusion seems to be easier to estimate than the worst collusion. This behavior suggests that the information about the collusion channel leaking from the observations is significantly higher for the minority collusion.

## 5.2 Decoding Performances

### 5.2.1 Thresholding

When the scores are compared to a threshold as detailed in Sec. 4.3, there are two types of error:

- False Alarm: at least one innocent is accused,
- False Negative: none of the colluders is accused.

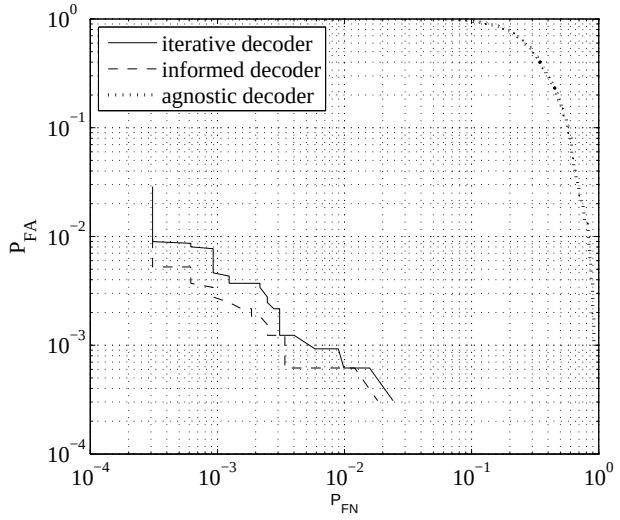
The performances of the decoder are evaluated by the probabilities of these two events, $\mathbb{P}_{FA}(T)$ and $\mathbb{P}_{FN}(T)$, measured experimentally by Monte Carlo. Fig. 5 shows the experimental ROC curve, obtained with $N = 3200$ realizations, as the decision threshold $T$ ranges from 0 to 1. Fig. 5(a) shows the resulting ROC when the colluders implement a Minority vote. In this case, the gap between the agnostic decoder and the iterative decoder is impressive, and moreover the latter performs closely to the optimal informed decoder. On the other hand, Fig. 5(b) shows the ROC when the colluders implement the Worst Case Attack. Clearly, the gap between the informed decoder and the agnostic decoder is greatly reduced. However, even in this case the performance of the proposed iterative decoder is very close to that of the informed decoder, and always better than the agnostic decoder.

Fig. 6 plots $\mathbb{P}_{FA}(T)$ vs. $\mathbb{P}_{FN}(T)$ from another perspective. Now, we fix $T^\star$ such that $\mathbb{P}_{FN}(T^\star) = 0.2$, ie. 20% of the time we miss all the colluders, and show the corresponding $\mathbb{P}_{FA}(T^\star)$ for different attacks and collusion size. Similar comments as for Fig. 5 apply in this case. Note that the rate of the code is $\log_2 n/m \approx 0.04$ bits per sample which exceeds the collusion channel capacity for $c \geq 4$. Therefore, big probabilities of errors are expected for $c \geq 4$. In real applications, the code should be longer.

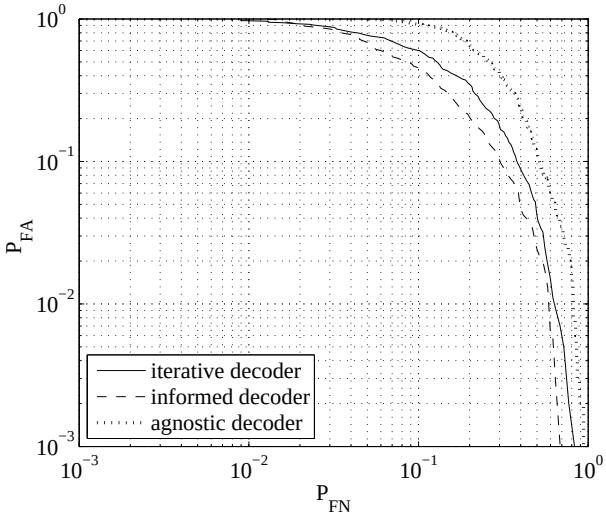
Finally, it is interesting to note that the huge variability in performance among different attacks, that has been observed in this set of experiments, is certainly correlated to the theoretical results shown in Tab. 1, where we can see that Minority has the biggest loss with regard to the optimal decoder.

### 5.2.2 Maximum score heuristic

Instead of a thresholding decision rule, we can use a maximum score heuristic, where only one user is accused: the one whose score is the largest. Thus, there is no threshold and the two errors mentioned above become now the same event.




(a) Minority




(b) Worst Case Attack

**Figure 5: Probabilities of false alarm and false negative with the iterative, informed, and agnostic decoders, for $c = 5$, $m = 300$, $n = 1000$ (a) Minority Attack and (b) Worst Case Attack.**

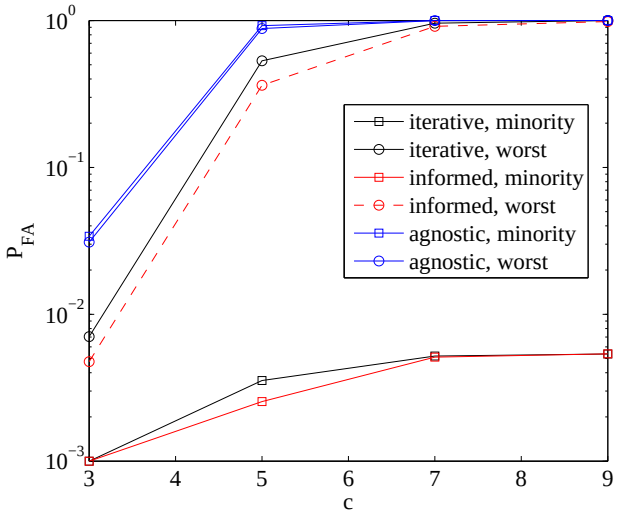



**Figure 6: Probability of false alarm when $\mathbb{P}_{FN}(T^\star) = 0.2$, $m = 250$, $n = 1000$.**

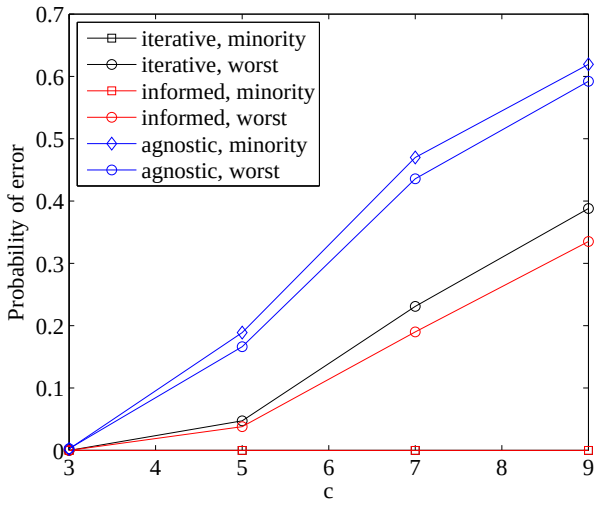



**Figure 7: Probability of error for the maximum score heuristic, with $m = 400$, $n = 1000$.**

Fig. 7 shows the probability of being wrong in accusing the user having the biggest score. The curves were obtained by Monte Carlo, in a set of $N = 1000$ experiments. The same behavior as in Sect. 5.2.1 can be noticed. Two comments are in order. First, the WCA is not the worst attack for this framework. The WCA is indeed defined as the collusion attack minimizing the mutual information between the symbols of the pirated copy and the user codeword. Thus, the WCA minimizes in general the upper bound on the accusation performances, but it might not be the worst attack for a given accusation process. Second, the curve of the iterative decoder against a minority attack is hidden behind the one for the informed decoder, ie. the probability of error is null.

## 6. CONCLUSION

The preliminary results of this paper show that the new proposed approach, based on joint channel estimation and decoding, opens the door to powerful decoding schemes for probabilistic traitor tracing codes. The main drawback of the E.-M. decoder resides in its complexity. The evaluation of the $\tilde{Q}$ function needs $O(nmc)$ elementary operations, and we must perform a maximization of several of these functions. This is the reason why experiments presented in this paper do not tackle long codes and many users. They clearly show that the E.-M. decoder performs significantly better than the agnostic decoder, and sometimes as good as the informed decoder. However, this experimental setup would not be applicable to scenarios where a huge number of users

is involved. Our future work will investigate less complex decoders and preprocessing techniques able to prune most of the users. We will investigate as well the fundamental limits in the estimation of the collusion model, in terms of the code length and the type of collusion attack.

On the other hand, the proposed decoding approach brings new possibilities that, to the best of our knowledge, have not been exploited yet in the decoding of traitor tracing codes. For instance, the approach can be extended to more general collusion models, other than the marking assumption used in this paper and in many others in the literature. Indeed, nothing prevents us from encompassing symbol erasures and random errors in the statistical model of the collusion. The difficulty is that the iterative decoder must estimate even more parameters. Yet, the agnostic decoder is not at all robust against erasures and especially random errors, since they violate the marking assumption. A larger gap between the agnostic decoder and the informed decoder can be expected in such case.

## 7. REFERENCES


[1] 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.

[2] O. Blayer and T. Tassa. Improved versions of Tardos' fingerprinting scheme. *Des. Codes Cryptography*, 48(1):79–103, 2008.

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

[4] A. Charpentier, F. Xie, C. Fontaine, and T. Furon. Expectation maximisation decoding of tardos probabilistic fingerprinting code. In N. Memon, editor, *Security, steganography and watermarking of multimedia contents*, San Jose, CA, USA, jan 2009. SPIE Electronic Imaging.

[5] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley series in Telecommunications, 1991.

[6] T. Furon, A. Guyader, and F. Cérou. On the design and optimization of tardos probabilistic fingerprinting codes. In *10th International Workshop on Information Hiding, IH'08*, Santa Barbara, California, USA, May 19-21 2008.

[7] T. Furon, L. Pérez-Freire, A. Guyader, and F. Cérou. Estimating the minimal length of tardos code. In *Information Hiding*, Darmstadt, Germany, jun 2009. Accepted.

[8] T. K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, November 1996.

[9] P. 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`.

[10] K. Nuida, S. Fujitsu, M. Hagiwara, T. Kitagawa, H. Watanabe, K. Ogawa, and H. Imai. An improvement of Tardos's collusion-secure fingerprinting codes with very short lengths. In *Proc. 9th International Workshop on Information Hiding, IH'07*, volume 4851 of *LNCS*, pages 80–89, Saint Malo, France, June 2007.

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

[12] 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.