Décodage EM du code de Tardos pour le *fingerprinting*

Ana Charpentier[1], Caroline Fontaine[2], Teddy Furon[1]

1. *INRIA, Centre Rennes – Bretagne Atlantique, Campus de Beaulieu 35042 Rennes, France*
2. *CNRS/Lab-STICC/CID et Télécom Bretagne/ITI[*], Technopôle Brest-Iroise CS 83818 - 29238 Brest Cedex 3, France*

RÉSUMÉ. *Cet article porte sur les codes traçants, qui ont pour objectifs d'identifier des utilisateurs impliqués dans la redistribution illégale de copies de documents, par exemple dans le contexte de la vidéo à la demande. Nous y présentons une amélioration de la phase d'accusation des codes de Tardos. Plus spécifiquement nous montrons comment l'optimiser en fonction de la stratégie d'attaque des pirates. Nous proposons également des moyens d'estimer à partir d'une copie le nombre d'attaquants qui se sont rassemblés pour la créer, ainsi que la stratégie qu'ils ont employée. Notre solution s'appuie sur un algorithme itératif à la EM, dans lequel une meilleure estimation de la stratégie permet une meilleure détection des attaquants, qui permet à son tour une meilleure estimation de la stratégie, etc.*

ABSTRACT. *This paper presents our recent work on multimedia fingerprinting, also known as traitor tracing. We focus on deriving a better accusation process for the well known Tardos codes, designed to identify dishonest users who redistributed illegally a piece of content. It appears that Tardos orginal decoding is very conservative: its performances are guaranteed whatever the collusion strategy. Indeed, major improvements stem from the knowledge of the collusion strategy. Therefore, this paper investigates how it is possible to learn and adapt to the collusion strategy. Our solution is based on an iterative algorithm a la EM, where a better estimation of the collusion strategy yields a better tracing of the colluders, which in return yields a better estimation of the collusion strategy etc.*

MOTS-CLÉS : *Watermarking, code anti-collusion, code de Tardos.*

KEYWORDS: *Watermarking, anti-collusion, Tardos code.*

DOI:10.3166/TS.27.127-146

[*] Ce travail a été effectué pendant que C. Fontaine était au CNRS/IRISA.

Extended abstract

This paper deals with *multimedia fingerprinting*, also known as *transactional watermarking*, *traitor tracing*, or *copy serialization*. The addressed problem is the scenario where a multimedia content server distributes personal copies of the same content to n different buyers. c dishonest users, called colluders in the sequel, mix their copies to forge a pirated content they will illegally redistribute. An accusation process aims at tracing back the colluders' identity by analyzing this pirated content. The pivotal techniques in this application are a robust watermarking technique, a short anti-collusion fingerprinting code, and their suitable association. In this paper, we focus on a famous fingerprinting code, the probabilistic Tardos code.

Symmetric Tardos fingerprinting code

We consider the probabilistic Tardos fingerprinting code through a variant proposed by Skoric *et al.* (Skoric *et al.,* 2008), that makes the computation of the accusation sum symmetric. The distributed codewords can be arranged as a $n \times m$ matrix $\mathbf{X}$, where the j-th user ($j = 1..n$) receives the codeword $\mathbf{X}_j = (X_{j1}, X_{j2}, \ldots, X_{jm})$. First step, m independent random variables $(p_1, \ldots, p_m)$ are drawn according to the probability density function $f : f(p) = \frac{1}{\sqrt{p(1-p)}}$. Each element of matrix $\mathbf{X}$ is independently and randomly drawn such as $\mathbb{P}(X_{ji} = 1) = p_i$.

At the accusation side, sequence $\mathbf{Y}$ is decoded from the pirated copy. A score S_j is first calculated for the j-th user: $S_j = \sum_{i=1}^{m} U(Y_i, X_{ji}, p_i)$. If $S_j > Z$, for a threshold Z, then the distributor regards user j as guilty. $U(Y, X, p)$ is defined as

$$U(Y,X,p) = \delta_X(Y) g_1(p(Y)) + (1 - \delta_X(Y)) g_0(p(Y))$$

with $g_0(p) = -\sqrt{\frac{p}{1-p}}$, $g_1(p) = \sqrt{\frac{1-p}{p}}$, $\delta_X(Y) = 1$ when $Y = X$, 0 otherwise, and $p(Y) = \delta_1(Y)p + \delta_0(Y)(1-p)$.

In his paper (Tardos, 2003), Tardos fixed the values of the three parameters f, g_0 and g_1 without further explanation. Recently, Furon, Cerou and Guyader (Furon *et al.*, 2008b) showed these functions are optimum in the general case; but there are two exceptions: if we know the size of the collusion (whatever the collusion strategy) $c \leq c_{max}$, there is a better function f (impact on the code initialization); if we know the collusion strategy, functions g_0 and g_1 are no longer optimal (impact on the accusation sum). However, this assumption about the collusion strategy is very critical. When functions g_1 and g_0 match the collusion strategy, the expectation of the score of the colluders is greater than the expectation of Tardos' scores. Yet, a mismatch has a dramatic effect.

General accusation design

Our approach changes the use of Tardos fingerprinting codes. First, in his original paper, the code length is constrained by the probabilities of errors (accusing an innocent, and missing a colluder). However, we argue that in real life the code length is more probably constrained by the size of the piece of content to be protected and the embedding rate of the chosen watermarking technique. Tardos'construction is so flexible that any length is manageable, and we will see that it is possible to adapt the accusation process to manage shorter codes in a quite efficient way.

Second, the original accusation is focused on one user at a time, deciding whether a given user is guilty or not. In our approach, we calculate the scores for all the users. Tardos accusation score is so simple that this does not bring any computation issue. Instead of comparing the score to a threshold Z (function of the probabilities of errors and the code length), we simply accuse the user who has the biggest score. Note that, although this user is the most likely to be guilty, there is no guarantee of not accusing an innocent. Contrary to Tardos, there is no threshold or code length to assess a probability of false alarm. However, in another work (Furon *et al.*, 2008a) we show that it is possible to accuse the biggest score while estimating the probability to be wrong.

Iterative accusation scores

We want to take advantage of optimized functions g_0 and g_1 as in (Furon *et al.,* 2008b), while decreasing the risk of mismatch. The collusion strategy is modeled by the probability $\mathbb{P}(Y_i = 1|\Sigma_i = \sigma_i)$, where $\Sigma_i = \sum_{j\in C} X_{ji}$, ie. the number of colluders' symbols equaling one at position i. We assume that the same strategy has been used for all the positions $1 \leq i \leq m$. To optimize the accusation sum, we therefore need to know the collusion strategy $\mathbb{P}(Y = 1|\Sigma = \sigma)$. This is done via an EM-like (Expectation-Maximisation) iterative process, sketched in figure 1 :

1) Initialization: We compute Tardos' accusation scores for all the users. The output is the sequence $\mathbf{S}^{(k)}$,$k = 1$ of n scores.

2) EM decoding: The sequence $\mathbf{S}^{(k)}$ amounts to a mixture of scores of innocents and colluders. A classical EM algorithm estimates the state, 'innocent' or 'colluder', of each scores. As input, it takes sequence $\mathbf{S}^{(k)}$. As output, it gives the estimated number of colluders $\hat{c}^{(k)}$ and the sequence $\hat{\mathbf{T}}^{(k)}$, where $\widehat{T}_j^{(k)}$ is the probability that the score $S_j^{(k)}$ of User j is the one of a colluder.

3) Kernel density estimator : With $\hat{c}^{(k)}$ and sequence $\hat{\mathbf{T}}^{(k)}$, we estimate the collusion strategy through probability $\hat{\mathbb{P}}^{(k+1)}(Y = 1|\Sigma = \sigma)$.

4) We provide optimized functions $g_0^{(k+1)}$ and $g_1^{(k+1)}$ matching the collusion strategy $\hat{\mathbb{P}}^{(k+1)}(Y = 1|\Sigma = \sigma)$. It gives us new scores, stored in sequence $\mathbf{S}^{(k+1)}$, and then we iterate proceeding with Step 2.

We noticed that there is no more difference between the results when the algorithm runs more than 2 times. The algorithm is forced to stop after 2 iterations. Note that the matched functions g_1 and g_0 derived in (Furon *et al.,* 2008b) ensure that the scores of the innocent are uncorrelated. For a big enough size of code m, the scores are almost Gaussian distributed (sum of iid random variables). Therefore, the innocent scores are deemed independent. This is a necessary condition for applying the EM algorithm at Step 2. However, this assumption is wrong for colluders scores. Our experimental test shows that this does not prevent the convergence of this algorithm for a small minority of colluders, ie. $c \ll n$.

Results and conclusion

Our results show that with these new functions, we can catch much more colluders than with the original ones. The iterative structure of our decoder allows to estimate the size and the strategy of the collusion. Its efficiency is experimentally shown especially when c is large (and in this case our scheme is really more efficient than previous ones). We can also see that the results are really different from one collusion strategy to another. The gap of performances is uneven. One remaining question is to end out why some collusion strategies are worse than others and, for a given collusion size c, what is indeed the worse one.

1. Introduction

Cet article porte sur le *multimedia fingerprinting*, connu aussi sous les noms de *transactional watermarking*, *traitor tracing*, ou *copy serialization*. Le but de ces méthodes est de lutter contre la redistribution non autorisée de documents. Un tel marquage de document a été pratiqué depuis longtemps : par le passé, les éditeurs ont parfois marqué leurs copies en modifiant légèrement une carte ou des données (les décimales non significatives sur les tables de logarithme de Néper en sont un exemple). Ils reconnaissaient ainsi une copie produite à partir d'un de leurs documents.

De nos jours, le scénario classique est le suivant : un serveur de vidéo à la demande distribue des copies d'un même contenu numérique à n acheteurs différents. Ces copies sont personnalisées et contiennent chacune un message qui les identifie, caché à l'aide d'une technique de tatouage. Pour ne pas ralentir la distribution des contenus, l'architecture souvent retenue est basée sur deux couches. Une technique de tatouage robuste joue le rôle de la couche « physique ». Avant la mise en vente, le contenu est découpé en une suite de blocs (par exemple, quelques secondes de vidéo), et chaque bloc est décliné en deux versions : dans l'une le symbole « 0 » est caché, dans l'autre le symbole « 1 ». Les blocs sont aussi chiffrés à l'avance. La deuxième couche dite code anti-collusion attribue une séquence de symboles unique à chaque utilisateur. Le serveur n'a plus qu'à envoyer à ce dernier les blocs correspondant à sa séquence ainsi que la clé de déchiffrement. Ainsi la personnalisation des copies ne provoque qu'une surcharge modeste pour le serveur. Des utilisateurs malhonnêtes, appelés *colluders*, utilisent leurs copies pour créer un faux qu'ils vont redistribuer de façon illégale. L'objectif est alors pour le distributeur

de contenus de retrouver l'identité de ces *colluders* par un processus d'accusation à partir du tatouage extrait de la copie pirate.

La première publication posant les bases du sujet est (Boneh *et al.*, 1998). Dans cet article, les auteurs posent un modèle de la collusion appelé *Marking Assumption* : la marque extraite de la copie illégale ne contient que les symboles qui apparaissaient dans les copies des *colluders* pour chaque position. Notre travail se place sous cette hypothèse : si les *colluders* possèdent tous un « 1 » pour un certain bloc, il leur est impossible de construire le même bloc contenant un « 0 », donc la copie pirate contiendra forcément un « 1 » dans ce bloc (à moins qu'ils ne réussissent à l'effacer). Boneh et Shaw proposent aussi une construction de code anti-collusion basée sur l'emboîtement d'un code interne et d'un code externe. Par la suite, ce schéma sera beaucoup étudié, en considérant différents types de codes internes et externes (Schaathun, 2004). Staddon et Stinson ont établi une classification des codes utilisés (Staddon *et al.*, 2001) insistant sur les liens entre les codes correcteur d'erreurs et le traçage de traîtres, voir aussi (Chor *et al.*, 2000).

D'un point de vue académique, l'efficacité d'un code anti-collusion est donnée par sa taille m pour un cahier des charges donné : n utilisateurs, c colluders respectant la Marking Assumption, et une probabilité d'accuser un innocent bornée par ε. Ces codes appartiennent à deux grandes familles : ceux qui permettent une traçabilité forte ($\varepsilon = 0$), et ceux qui n'ont qu'une traçabilité faible ($\varepsilon > 0$). Les premiers sont en général très longs, difficiles à implémenter et demandant une grosse puissance de calcul lors de la phase de décodage. C'est pourquoi dans la pratique, on s'intéresse à ceux ayant une traçabilité faible. C'est à cette famille qu'appartiennent les codes de Tardos.

En 2003, G. Tardos est le premier à exhiber des codes traçants binaires dont la longueur atteint la borne inférieure théorique dite de Peikert : $m = O(c^2 log(n/\varepsilon))$ (Peikert *et al.*, 2003). Sa construction est un processus probabiliste des plus faciles à implémenter, où les probabilités d'erreur d'accusation (risque d'accuser un innocent, risque de n'identifier aucun pirate) sont contrôlées. Skoric *et al.* ont proposé une version symétrique de ces codes et ont étendu l'application à un alphabet q-aire (Skoric *et al., 2008)*. Furon *et al.* (Furon *et al.*, 2008b) ont démontré que les fonctions d'accusation sont optimales dans un contexte général, mais aussi qu'il existe des fonctions plus efficaces si le décodeur a des informations sur l'attaque qui a été réalisée. Ici, nous continuons dans cette direction en proposant des mécanismes d'estimation dynamique de la stratégie des pirates ainsi que de leur nombre, et des fonctions d'accusation optimisées pour cette stratégie.

2. Limites des études précédentes

Avant d'aller plus loin dans l'explication de notre solution, nous présentons brièvement les codes binaires de Tardos et l'accusation symétrique de Skoric. Pour plus de détails, nous renvoyons le lecteur vers les articles cités dans l'introduction. Soit n le nombre total d'utilisateurs, et m la longueur du code. Les mots de code distribués forment une matrice $n \times m$ binaire $\mathbf{X}$. L'utilisateur j se voit associé le mot binaire $\mathbf{X}_j = (X_{j1}, X_{j2}, \ldots, X_{jm})$.

– **Initialisation** : pour générer cette matrice, m nombres réels $p_i \in [t, 1-t]$ sont distribués selon une fonction de densité de probabilité $f(p) : [t, 1-t] \to \mathbb{R}^+$. t est un réel positif tel que $ct \ll 1$. On note $\mathbf{p} = (p_1, \ldots, p_m)$.

– **Construction** : chaque élément de la matrice $\mathbf{X}$ est ensuite indépendamment tiré en suivant la probabilité $\mathbb{P}(X_{ji} = 1) = p_i$.

Chacun de ces n mots de code est caché dans la copie délivrée à l'utilisateur associé comme expliqué dans l'introduction.

– **Mise au secret** : $\mathbf{X}$ et $\mathbf{p}$ sont les secrets du code, partagés avec l'accusation. Les colluders peuvent découvrir leur mots de code respectifs sans porter atteinte à la sécurité du code. En revanche, une hypothèse de travail est qu'ils ne connaissent pas le mot de code d'un utilisateur innocent.

– **Accusation** : on extrait la séquence $\mathbf{Y}$ de la copie pirate. Afin de savoir si l'utilisateur j est impliqué dans la production de la contrefaçon, on calcule un score d'accusation S_j. Si ce score est supérieur à un certain seuil Z, alors on considère l'utilisateur j comme coupable. Une variante est d'accuser l'utilisateur le plus probablement coupable, *ie*. celui qui a le plus gros score. Le calcul des scores repose sur quatre fonctions d'accusation, qui évaluent la corrélation entre la séquence $\mathbf{X}_j$, associée à l'utilisateur j, et la séquence extraite $\mathbf{Y}$:

$$S_j = \sum_{i=1}^{m} U(Y_i, X_{ji}, p_i), \qquad [1]$$

avec les fonctions d'accusation

$$U(1,1,p) = g_{11}(p) > 0, \qquad U(0,0,p) = g_{00}(p) > 0,$$

$$U(0,1,p) = g_{01}(p) < 0, \qquad U(1,0,p) = g_{10}(p) < 0.$$

Quelques commentaires pour comprendre le code de Tardos :

– Le code est mi-dense mi-creux. Il est dense pour les indices i tels que $p_i \approx 0.5$ au sens où ces colonnes de $\mathbf{X}$ comportent en espérance autant de symboles « 0 » que de « 1 ». Il est creux pour les indices i tels que p_i est proche de 0 (ou 1) au sens où ces colonnes de $\mathbf{X}$ comportent presque que des symboles « 0 » (respectivement, presque que des « 1 »). Pour une densité de code donnée, il existe une attaque au pire cas. Les valeurs de $\mathbf{p}$ étant secrètes, les colluders ne peuvent adapter leur attaque à la densité du code. Ils n'ont donc qu'une stratégie globale alors que le code possède une pluralité de densités.

– Pour un utilisateur donné, un test d'hypothèse est réalisé. Le score d'accusation est comparable à un faisceau de preuves. Avoir le même symbole que celui retrouvé dans la copie pirate (*i.e*. $Y_i = X_{ji}$) n'est pas une preuve irréfutable de la culpabilité de l'utilisateur, mais cela tend à corroborer cette hypothèse : la somme d'accusation est augmentée d'un poids positif $g_{Y_i X_{ji}}(p_i)$. À l'inverse, un symbole différent n'est pas une preuve irréfutable de son innocence, mais cela tend à l'innocenter, d'où un poids négatif. L'amplitude de l'accusation dépend de la rareté de correspondance entre la valeur de la copie pirate et celle de l'utilisateur. On sent bien par exemple que si $Y_i = 0$ et p_i est proche de 1, alors très peu d'utilisateurs ont un symbole « 0 » à cet indice : par conséquent, le poids sera positif et de forte amplitude pour ces derniers.

2.1. Comment choisir la fonction f et les fonctions d'accusation de façon adéquate ?

Dans l'article original (Tardos, 2003), Tardos donne les trois fonctions f, g_{10} et g_{11} suivantes (g_{00} et g_{01} étaient initialement égales à 0) et prouve qu'avec ce choix, les performances de l'accusation ne dépendent pas de la stratégie des colluders :

$$f(p) = \frac{1}{\pi\sqrt{p(1-p)}},\ g_{11}(p) = \sqrt{\frac{1-p}{p}},\ \text{et}\ g_{10}(p) = \sqrt{\frac{p}{1-p}}. \qquad [2]$$

Mais il n'a pas expliqué comment il avait trouvé ces fonctions. Ces choix ont été conservés par Skoric *et al.*, qui ont proposé une accusation « symétrique », donnant à g_{00} et g_{01} des valeurs non nulles (Skoric *et al.*, 2008) :

$$g_{11}(p) = g_{00}(1-p) = -g_{01}(p) = -g_{10}(1-p) = \sqrt{\frac{1-p}{p}}. \qquad [3]$$

Une relecture statisticienne apporte une nouvelle interprétation. La stratégie de la collusion n'est pas connue à l'accusation : c'est un paramètre de nuisance. L'idée de G. Tardos est de créer un test basé sur une quantité pivot indépendante du paramètre de nuisance. Les fonctions choisies assurent cette indépendance uniquement pour les moments d'ordre 1 et 2 :

$$\text{Innocent :}\quad E[S_j] = 0, \qquad E[S_j^2] = m \qquad [4]$$

$$\text{Colluder :}\quad E[S_j] = \frac{2m\pi}{c}, \qquad E[S_j^2] = m \qquad [5]$$

Grâce à la borne de Markov-Chebyshev, les probabilités d'accuser à tort et de rater un coupable sont majorées et inférieures aux niveaux donnés dans le cahier des charges pour une longueur de code $m = O(c^2 \log(n/\varepsilon))$ [1].

Plus récemment, (Furon *et al.*, 2008b) ont donné des améliorations si on relâche certaines hypothèses : connaître la taille maximum de la collusion c_{max} lors de la génération du code aide à déterminer une meilleure fonction f, quelle que soit la stratégie ; connaître la stratégie de la collusion aide à déterminer de meilleures fonctions lors de l'accusation. Lorsque les fonctions d'accusation correspondent à la stratégie de collusion, l'espérance des scores des colluders est supérieure à l'espérance pour le score de Tardos. Mais en revanche, lorsque qu'il y a une mauvaise correspondance, les effets peuvent être désastreux.

Une alternative à l'accusation très conservatrice de G. Tardos est ouverte. Cependant, l'hypothèse admise dans (Furon *et al.*, 2008b) sur la stratégie de collusion est critiquable : les auteurs ont optimisé les fonctions d'accusation en fonction de la stratégie de collusion mais ont conservé une contrainte (Eq. (14) de (Furon *et al.*,

1. Nous avons omis de nombreux détails, la démonstration est en fait bien plus technique (cf. (Blayer *et al.*, 2008).)

2008b) [qui peut être réécrite avec nos notations comme $(1-p)g_{00}(p) = pg_{11}(p))$] liée à l'indépendance entre la variance des scores des innocents et la stratégie des colluders. Ici, nous relâchons cette hypothèse, en ne conservant que le minimum de contraintes possible sur les fonctions d'accusation et en les optimisant en utilisant une estimation de la stratégie des colluders. Pour mesurer les performances de notre solution, nous la comparons avec la version symétrique des codes de Tardos faite par (Skoric *et al.*, 2008) ainsi qu'avec l'optimisation de (Furon *et al.*, 2008b).

2.2. Comment réduire la taille du code ?

L'une des propriétés les plus intéressantes des codes de Tardos est sa longueur minimale. Cette longueur est déterminée par la taille maximum de la collusion et les probabilités d'erreur (d'accuser un innocent, et de ne pas trouver de colluder). Malgré cela, dans un schéma réel on est amené à utiliser des codes plus courts, en ce sens où la contrainte vient de la taille du contenu à protéger et du taux d'insertion de la technique de marquage choisie. Dans ce cas, il est intéressant d'utiliser les codes de Tardos pour leur implémentation facile, et d'adapter le processus d'accusation pour gérer la longueur et conserver une bonne efficacité de traçage.

Nous proposons ici d'utiliser un algorithme d'accusation itératif *a la* EM (Expectation-Maximization) afin d'obtenir une meilleure accusation même lorsque la longueur du code est plus courte que celle préconisée par G. Tardos.

2.3. Comment estimer la pertinence de l'accusation ?

Dans le processus original, chaque utilisateur j est testé indépendamment et on n'a donc pas besoin de calculer tous les scores pour rendre un verdict. Cet utilisateur est considéré comme un *colluder* si son score est supérieur au seuil Z. Le seuil est choisi pour garantir une certaine probabilité de fausse alarme. Ainsi, lorsqu'on accuse un utilisateur on a une borne supérieure de la probabilité d'erreur, mais on n'a pas d'estimation précise de cette erreur. On adopte ici une autre stratégie, en calculant les scores de tous les utilisateurs et en accusant celui qui a le plus grand. On peut noter que bien que cet utilisateur soit celui qui est le plus sûrement coupable, il n'y a pas de garantie de ne pas accuser un innocent. On peut ou bien comparer ce score à un seuil afin de refuser de porter une accusation lorsque même le score maximal est trop faible, ou bien estimer expérimentalement la probabilité d'accuser à tort (Furon *et al.*, 2008a).

3. Une estimation itérative de la stratégie

En suivant les pas de (Furon *et al.*, 2008), notre objectif est d'optimiser les fonctions d'accusation en fonction de la stratégie des *colluders*. Nous procédons en deux étapes : d'abord nous estimons la stratégie, puis nous optimisons les fonctions d'accusation. Ce procédé est itéré, chaque itération tirant profit d'une nouvelle estimation de l'ensemble des *colluders via* Expectation-Maximization (EM). Nous avons remarqué que les résultats évoluent peu après 2 itérations. L'algorithme est donc forcé à s'arrêter au bout de 2 itérations.

On modélise la stratégie de la collusion par l'ensemble des probabilités $\{\mathbb{P}(Y_i = 1|\Sigma_i = \sigma_i), \sigma_i = 0..c\}_{i=1..m}$; la variable aléatoire $\Sigma_i = \sum_{j \in \mathcal{C}} X_{ji}$ correspond aux nombre de mots des *colluders* ayant un ' 1 ' à la position i. On admet que la même stratégie a été utilisée pour chaque position $1 \leq i \leq m$. Afin d'alléger les formules, nous omettrons l'indice i qui indique la position considérée et appellerons $\boldsymbol{\theta}$ le modèle de collusion : $\boldsymbol{\theta} = \{\mathbb{P}(Y = 1|\Sigma = \sigma), \sigma = 0..c\}$. Nous décrivons maintenant les étapes du processus d'estimation itérative en détail, en utilisant les notations allégées. Le processus est illustré par la figure 1.

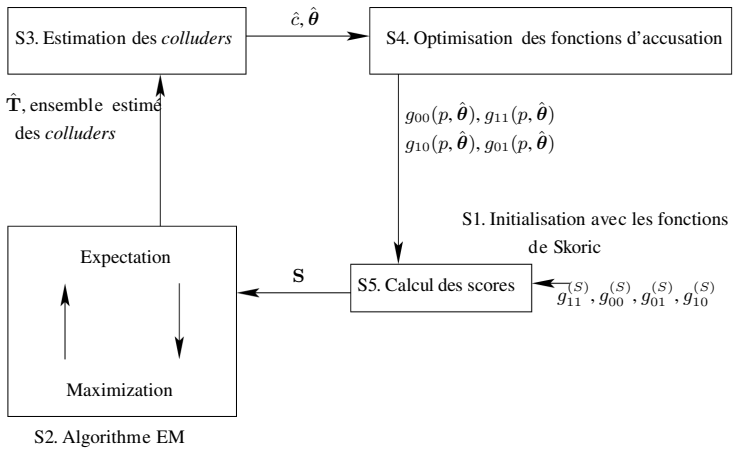


Figure 1. Calcul des nouvelles fonctions d'accusation

S1. Initialisation : nous calculons tous les scores d'accusation avec les fonctions d'accusation de B. Skoric *et al* : $g_{11}^{(S)}(p) = g_{00}^{(S)}(1-p) = -g_{01}^{(S)}(p) = -g_{10}^{(S)}(1-p) = \sqrt{\frac{1-p}{p}}$. Ces scores sont placés dans le vecteur $\mathbf{S}$.

S2. Décodage EM : la séquence $\mathbf{S}$ contient un mélange de scores d'innocents et de *colluders*. Un algorithme EM classique estime le statut, « innocent » ou « colluder », de chaque utilisateur. EM prend en entrée le vecteur $\mathbf{S}$ et les moyennes et variances théoriques des scores des innocents et *colluders*. En sortie, on a le vecteur $\hat{\mathbf{T}}$; $\widehat{T}_j$ correspond à l'estimation de la probabilité que le score S_j de l'utilisateur j soit celui d'un *colluder*. Un exemple classique de l'algorithme EM est le mélange de gaussiennes où il s'agit de séparer les échantillons de plusieurs (deux dans notre cas) distributions dont on sait qu'elles sont gaussiennes mais de variances et espérances à déterminer (Delyon, 2010, Sec. II.7). Nous l'appliquons pour séparer les distributions des scores des innocents, d'une part et les scores des *colluders*, d'autre part. Nous avons choisi d'initialiser l'algorithme avec des données pertinentes, ces paramètres d'espérance et de variance sont donc remplis avec les valeurs données théoriquement pour les scores de Tardos. Le nombre de *colluders* c est initialisé à 2.

S3. Avec $\hat{\mathbf{T}}$ et $\mathbf{S}$, on estime la taille de la collusion, notée $\hat{c}$, ainsi que sa stratégie, notée $\hat{\boldsymbol{\theta}}$.

S4. En considérant $\hat{\boldsymbol{\theta}}$ on optimise les fonctions d'accusation $g_{00}(p,\hat{\boldsymbol{\theta}})$, $g_{11}(p,\hat{\boldsymbol{\theta}})$, $g_{10}(p,\hat{\boldsymbol{\theta}})$, et $g_{01}(p,\hat{\boldsymbol{\theta}})$.

S5. On calcule les nouveaux scores, conservés dans le vecteur $\mathbf{S}$. On revient alors à l'étape S2 pour itérer.

Les fonctions d'accusation optimisées obtenues dans (Furon *et al.*, 2008b) sont obtenues sous l'hypothèse que les scores des innocents ne sont pas corrélés entre eux. De fait, pour une valeur de m assez grande, les scores sont considérés comme suivant des distributions gaussiennes (somme de variables aléatoires i.i.d.). C'est pourquoi les scores des innocents seront supposés indépendants. C'est une condition nécessaire pour appliquer l'algorithme EM à l'étape 2. Cette affirmation est fausse pour les scores des *colluders*, mais nos résultats expérimentaux montrent que cela n'empêche pas la convergence de cet algorithme pour un petit nombre de *colluders* $c \ll n$.

Nous allons maintenant détailler les étapes clés de notre algorithme.

3.1. Estimation de la stratégie : étape S3

On considère les scores $\mathbf{S}$ et les probabilités associées dans le vecteur $\hat{\mathbf{T}}$; $\widehat{T}_j = 1$ signifie que l'utilisateur j est un *colluder*, $\widehat{T}_j = 0$ signifie qu'il est innocent. Dans la pratique, et les $\widehat{T}_j$ que nous obtenons ne sont pas binaires, ils sont une probabilité que l'utilisateur j ait participé à la création de la copie pirate. Nous estimons la taille de la collusion par $\hat{c} = \lceil \Sigma_{j=0}^{n} \widehat{T}_j \rceil$. On considère ensuite les $\hat{c}$ utilisateurs correspondant aux plus grandes probabilités $\widehat{T}_j$, et on utilise leurs séquences pour calculer le modèle de collusion $\hat{\boldsymbol{\theta}}$. Pour chaque position i, on calcule σ_i comme somme des X_{ji} pour chaque utilisateur j dans l'ensemble estimé des *colluders*. Pour chaque valeur possible de $0 \leqslant \sigma_i \leqslant c$, on calcule la moyenne des éléments de $\mathbf{Y}$ correspondants.

3.2. Optimisation de l'accusation : étape S4

Les nouvelles fonctions d'accusation sont obtenues par une optimisation sous contraintes, pour une collusion estimée donnée $\hat{\boldsymbol{\theta}}$. On note μ_{Inn} et ν_{Inn} (resp. μ_{Coll} et ν_{Coll}) l'espérance et la variance de la distribution des scores des utilisateurs innocents (resp. *colluders*), et $\kappa(S_j,S_k)$ la covariance entre les scores des utilisateurs j et k. De par la construction du code, les symboles sont i.i.d. d'un index à l'autre. Ceci implique, avec [1], que les statistiques des scores sont linéaires en m :

$$\mu_{Inn} = m\tilde{\mu}_{Inn}, \qquad \nu_{Inn} = m\tilde{\nu}_{Inn}\,, \tag{6}$$

$$\mu_{Coll} = m\tilde{\mu}_{Coll}, \qquad \nu_{Coll} = m\tilde{\nu}_{Coll}. \tag{7}$$

On résume les contraintes principales :

– les scores des innocents sont centrés : $\tilde{\mu}_{Inn} = 0$,

– les scores des innocents sont normalisés : $\tilde{v}_{Inn} = 1$,

– deux utilisateurs innocents ont des scores indépendants, ce qui se traduit sous les affirmations gaussiennes, par $\kappa(S_j, S_k) = 0$.

Ce sont les mêmes contraintes que dans (Furon *et al.*, 2008b), excepté qu'ici on ne contraint pas la variance des scores des *colluders*.

La distance de Kullback-Leibler mesure la « distance » entre les distributions des scores des *colluders* et des innocents. La théorie de la détection nous dit qu'elle doit être la plus grande possible afin d'assurer une bonne séparation entre les innocents et les *colluders*, ce qui donne des verdicts fiables. Comme nous avons déjà considéré que les scores des *colluders* suivent une distribution normale $\mathcal{N}_{Coll}$, et que les scores des utilisateurs innocents une distribution normale $\mathcal{N}_{Inn}$, la distance de Kullback-Leibler entre deux distributions normales satisfaisant $\tilde{\mu}_{Inn} = 0$ et $\tilde{v}_{Inn} = 1$ est la suivante :

$$D_{KL}(\mathcal{N}_{Coll}, \mathcal{N}_{Inn}) = \frac{1}{2}\left(m\tilde{\mu}_{Coll}^2 - \log(\tilde{v}_{Coll}) + \tilde{v}_{Coll} - 1\right). \quad [8]$$

Comme m est très grand, le terme prépondérant de la somme est $m\tilde{\mu}_{Coll}^2$. Notre objectif est de maximiser $\tilde{\mu}_{Coll}$ sous les contraintes $\mu_{Inn} = 0$, $\kappa(S_j, S_k) = 0$, et $\tilde{v}_{Inn} = 1$.

Considérant ces conditions, les fonctions qui maximisent $\tilde{\mu}_{Coll}$ sont :

$$g_{11}(p,\boldsymbol{\theta}) = \frac{1}{2\lambda}\frac{1-p}{q(p,\boldsymbol{\theta})}A(p,\boldsymbol{\theta}), \qquad g_{00}(p,\boldsymbol{\theta}) = \frac{1}{2\lambda}\frac{p}{1-q(p,\boldsymbol{\theta})}A(p,\boldsymbol{\theta}),$$

$$g_{10}(p,\boldsymbol{\theta}) = -\frac{p}{1-p}g_{11}(p,\boldsymbol{\theta}), \qquad g_{01}(p,\boldsymbol{\theta}) = -\frac{1-p}{p}g_{00}(p,\boldsymbol{\theta}) \quad [9]$$

avec :

$$\lambda = \frac{1}{2}\sqrt{\mathbb{E}_p\left[A^2(p,\boldsymbol{\theta})\frac{p}{q(p,\boldsymbol{\theta})}\frac{1-p}{1-q(p,\boldsymbol{\theta})}\right]}, \quad [10]$$

$$q(p,\boldsymbol{\theta}) = \mathbb{P}(Y=1|P=p,\boldsymbol{\theta}), \quad [11]$$

$$A(p,\boldsymbol{\theta}) = \mathbb{P}(Y=1|X=1,P=p,\boldsymbol{\theta}) - \mathbb{P}(Y=1|X=0,P=p,\boldsymbol{\theta}). \quad [12]$$

Les résultats nous permettent de calculer l'expression du μ_{Coll} maximisé :

$$\tilde{\mu}_{Coll} = \sqrt{\mathbb{E}_p\left[A^2(p,\boldsymbol{\theta})\frac{p}{q(p,\boldsymbol{\theta})}\frac{1-p}{1-q(p,\boldsymbol{\theta})}\right]}. \quad [13]$$

Preuve. On maximise cette expression en utilisant un Lagrangien $J(g_{11}, g_{00}) = \tilde{\mu}_{Coll} - \lambda(\tilde{\nu}_{Inn} - 1)$. Voir les détails en annexe.

3.3. Résultats théoriques

Lors des expériences, nous considérons différentes stratégies utilisées par les *colluders* pour créer $\mathbf{Y}$:

Uniforme les *colluders* choisissent au hasard un symbole parmi leurs copies :

$$\mathbb{P}(Y = 1 | \Sigma = \sigma) = \sigma / c \, ;$$

Majorité les *colluders* choisissent le symbole le plus fréquent :

$$\mathbb{P}(Y = 1 | \Sigma = \sigma) = 1 \text{ si } \sigma > c/2, \quad 0 \text{ sinon} \, ;$$

Minorité les *colluders* choisissent le symbole le moins fréquent :

$$\mathbb{P}(Y = 1 | \Sigma = \sigma) = 0 \text{ si } \sigma > c/2, \quad 1 \text{ sinon} \, ;$$

Tableau 1. Valeurs de $cm\tilde{\mu}_{Coll}/\sqrt{\tilde{\nu}_{Inn}}$ *obtenues après optimisation des fonctions d'accusation pour* $m = 100$, $c = 3{,}4{,}5$. *Entre parenthèses sont données celles obtenues par Furon* et al. *Rappelons qu'avec les fonctions de Skoric* et al *on a la valeur* 64 *dans tous les cas*

		Stratégie des *colluders*				
c	accusation	Uniforme	Majorité	Minorité	All1	AllO
3	Uniforme	**98 (71)**	106 (80)	100 (53)	97 (66)	97 (66)
	Majorité	96 (67)	**110 (84)**	100 (34)	95 (59)	95 (59)
	Minorité	81 (50)	59 (38)	**112 (75)**	89 (56)	89 (56)
	All1	83 (69)	88 (73)	88 (62)	**114 (68)**	84 (68)
	All0	83 (69)	88 (73)	88 (62)	84 (68)	**114 (68)**
4	Uniforme	**98 (71)**	106 (80)	105 (44)	99 (62)	99 (62)
	Majorité	96 (67)	**110 (84)**	105 (17)	97 (50)	97 (50)
	Minorité	61 (34)	25 (15)	**128 (91)**	88 (53)	88 (53)
	All1	79 (65)	83 (63)	88 (72)	**121 (67)**	87 (67)
	All0	79 (65)	83 (63)	88 (72)	87 (67)	**121 (67)**
5	Uniforme	**98 (71)**	110 (83)	110 (33)	100 (58)	100 (58)
	Majorité	94 (63)	**120 (93)**	113 (-22)	98 (35)	98 (35)
	Minorité	37 (19)	-20 (-17)	**155 (121)**	82 (52)	82 (52)
	All1	77 (59)	83 (47)	90 (90)	**128 (69)**	90 (69)
	All0	77 (59)	83 (47)	90 (90)	90 (69)	**128 (69)**

All1 si ils ont au moins un « 1 », les *colluders* mettent un « 1 » :

$$\mathbb{P}(Y = 1|\Sigma = \sigma) = 1 \text{ si } \sigma \neq 0 \,;$$

All0 si ils au moins un « 0 », les *colluders* mettent un « 0 » :

$$\mathbb{P}(Y = 1|\Sigma = \sigma) = 0 \text{ si } \sigma \neq c.$$

Ces stratégies sont conformes à la « marking assumption » car on a toujours $\mathbb{P}(Y = 1|\Sigma = 0) = 0$ et $\mathbb{P}(Y = 1|\Sigma = c) = 1$.

On calcule le ratio $cm\tilde{\mu}_{Coll}/\sqrt{\tilde{v}_{Inn}}$ obtenu avec les fonctions optimisées du théorème 3.2, et on les compare dans le tableau 1 aux précédents résultats de (Furon *et al.*, 2008b), pour lesquels $\tilde{v}_{Coll}$ était contrainte à être égale à 1. On voit que nos fonctions d'accusation sont, comme espéré, plus efficaces pour la stratégie des *colluders* pour laquelle elles ont été calculées. On voit aussi qu'elles sont plus efficaces que celles de (Furon *et al.*, 2008b) dans tous les cas, y compris lorsque les stratégies ne coïncident pas.

3.4. Résultats expérimentaux

Pour tous les résultats qui suivent, les vecteurs $\mathbf{p}$ ont été générés avec la fonction f donnée par Tardos. Suite aux travaux de Blayer (Blayer *et al.*, 2008) nous avons choisi d'augmenter la valeur du paramètre t par rapport à celle donnée par Tardos. Dans ce qui suit, on a $t = 1/100$. Les résultats se présentent sous la forme suivante : nous observons les k utilisateurs ayant les plus grands scores, et pour chacun, nous donnons la probabilité d'être un *colluder*. Sur la figure 2(a), on voit ainsi que le plus grand score correspond à celui d'un *colluder* dans 100 % des cas pour toutes les stratégies observées alors que le cinquième plus grand score est celui d'un *colluder* dans seulement 60 % des cas pour la stratégie majorité, par exemple. Ces probabilités sont estimées par des simulati ons numériques de type Monte-Carlo.

3.4.1. Première expérience : test des fonctions

Nous regardons d'abord le comportement des fonctions, en supposant que nous avons correctement estimé la stratégie ainsi que le nombre de colluders.

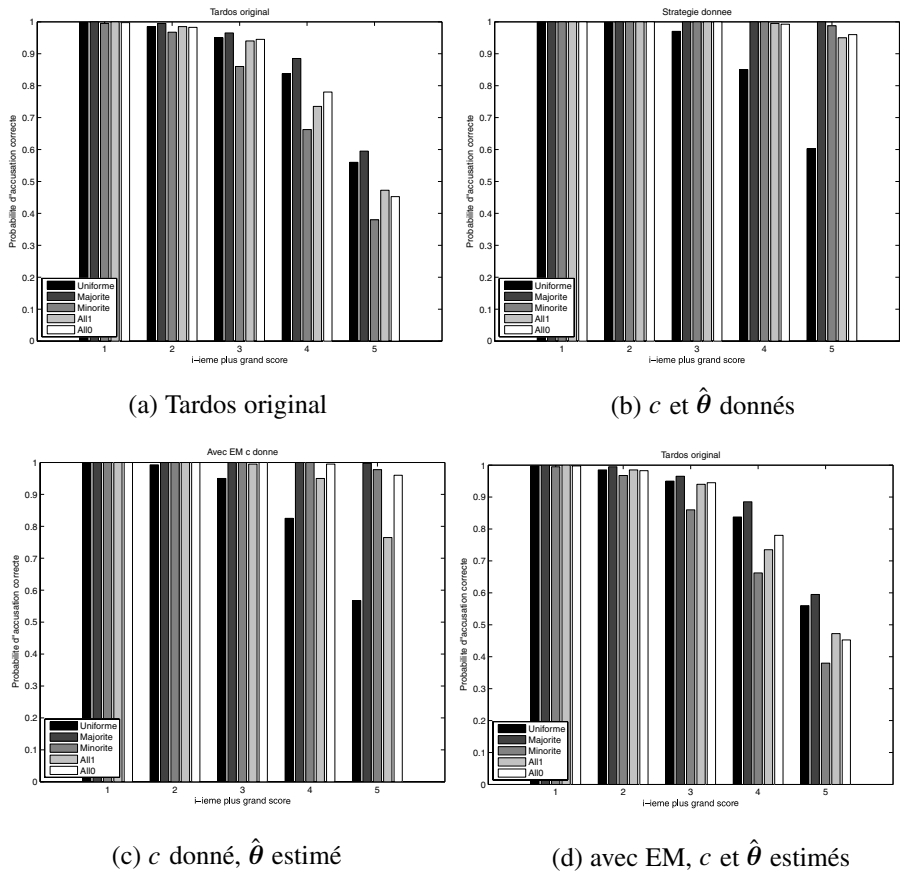
Les figures 2(b) et 3(b) montrent que ces nouvelles fonctions donnent de meilleures performances. Deux remarques sont à faire : les performances varient beaucoup suivant la stratégie de la collusion. Ceci est tout à fait normal. Contrairement à Tardos qui donne une accusation la plus invariante possible à ce paramètre de nuisance, nous choisissons au contraire d'exploiter sa connaissance. Du coup, il y a des stratégies de collusion qui sont pires que d'autres. Le prix à payer pour être invariant à la stratégie de collusion est donc une perte des performances plus ou moins grande suivant la nocivité de la stratégie.

Deuxièmement, l'amélioration par rapport au décodage de Tardos semble plus grande lorsque le nombre de *colluders* croît. Les deux expériences, $c = 5$ et $c = 8$, étant réalisées pour $m = 1\ 000$, cela montre que le code de Tardos est trop court pour

lutter contre une collusion de taille $c = 8$, sauf si on exploite la connaissance de la stratégie de collusion. Autrement dit, cette idée permettrait de réduire la taille des codes pour un niveau de performances donné. En réalité, ceci est faux car rien ne dit que les *colluders* vont choisir une stratégie peu dangereuse. Si on veut garantir un niveau de performance quelle que soit la collusion, alors il ne faut retenir que la stratégie la plus nocive. Dans nos expériences, c'est la stratégie « uniforme », et l'écart de performances par rapport au décodage de Tardos n'est pas si grand que cela. Autrement dit, le code de Tardos conserve la même longueur m, mais si les *colluders* ne sont pas les plus malicieux, ils seront plus sûrement trouvés avec notre décodage.

3.4.2. Seconde expérience : test de l'estimateur de stratégie

Nous supposons maintenant que le décodeur connaît c et il doit estimer la stratégie. Le fait de connaître le nombre de *colluders* permet d'améliorer la phase d'initialisation de notre schéma : nous pouvons, lors de la première itération, calculer



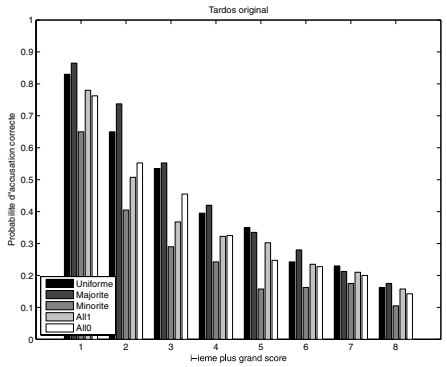
(a) Tardos original

(b) c et $\hat{\boldsymbol{\theta}}$ donnés

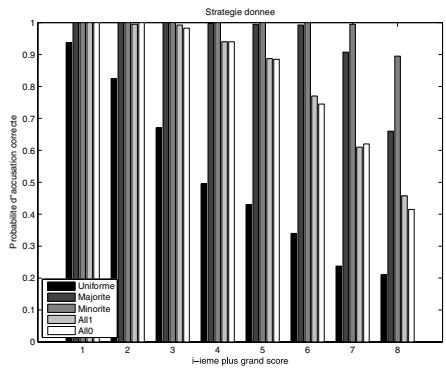
(c) c donné, $\hat{\boldsymbol{\theta}}$ estimé

(d) avec EM, c et $\hat{\boldsymbol{\theta}}$ estimés

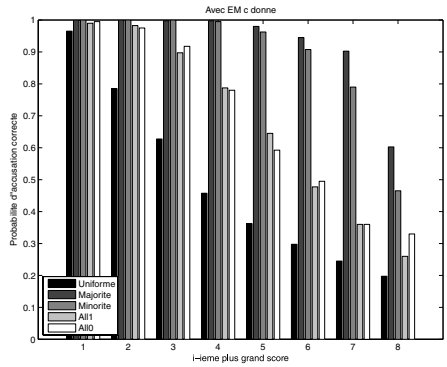
Figure 2. Probabilité d'accuser correctement le k-ième plus grand score, pour $k \in \{1,\ldots,5\}$. $m = 1000$, $c = 5$, $n = 5000$. 400 expériences réalisées



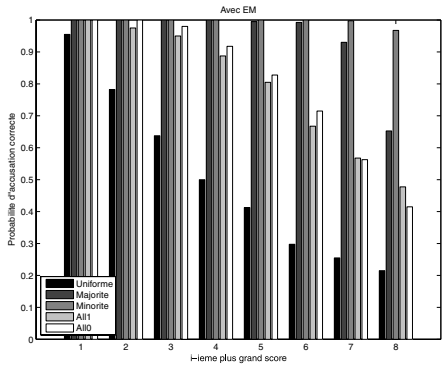
(a) Tardos original



(b) c et $\hat{\boldsymbol{\theta}}$ donnés



(c) c donné, $\hat{\boldsymbol{\theta}}$ estimé



(d) avec EM, c et $\hat{\boldsymbol{\theta}}$ estimés

Figure 3. Probabilité d'accuser correctement le k-ième plus grand score, pour $k \in \{1,\ldots,8\}$. $m = 1000$, $c = 8$, $n = 5000$. 400 expériences réalisées

les probabilités T_j sans algorithme EM car nous connaissons les distributions des scores : des Gaussiennes $G(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp -\frac{(x-\mu)^2}{2\sigma^2}$ d'espérance et de variance données par [5] et [4]. Ainsi :

$$\widehat{T}_j = \frac{cG(S_j;\mu_{Coll},\sigma_{Coll})}{cG(S_j;\mu_{Coll},\sigma_{Coll}) + (n-c)G(S_j;\mu_{Inn},\sigma_{Inn})}.$$

On voit sur les figures 2(c) et 3(c) que cette méthode donne de moins bons résultats. L'estimation de la stratégie de collusion est imparfaite, et cela se répercute sur les performances de l'accusation. Les stratégies déterministes (« Majorité »,

« Minorité », « All1 », « All0 »), au sens où y_i est une fonction déterministe de σ_i, sont plus difficiles à estimer que pour la stratégie « uniforme ». Ceci est un moindre mal puisque nos fonctions d'accusation étaient très efficaces contre ce type de stratégie.

3.4.3. Troisième expérience : test de l'estimateur du nombre de colluders

Maintenant, le décodeur ignore *a priori* la valeur de c. C'est l'algorithme EM qui va en fournir une estimation. La précision de cette estimation apparaît comme fortement liée au quotient c/n. Nous avons choisi de forcer l'estimateur à trouver un nombre de colluders entre $c_{\min} = 2$ et $c_{\max} = 10$. Pour la stratégie « Uniforme », l'importance de c est nulle car les fonctions d'accusation sont les mêmes pour deux tailles de collusion différentes. En revanche, elles sont au contraire très dépendantes de c pour les stratégies déterministes.

Nous obtenons parfois de meilleurs résultats que lorsque c est donné et que lorsque la stratégie est donnée. Cela vient du fait que le score est optimisé dans l'annexe *pour une valeur c donnée*, et que si $\hat{c} \neq c$, nous ne sommes pas sûrs de remplir la contraire $\tilde{\nu}_{Inn} = 1$ ni même d'avoir maximisé $\tilde{\mu}_{Coll}$. Ces deux valeurs dépendent de la vraie stratégie de collusion $\boldsymbol{\theta}$ qui restera toujours inconnue pour le décodeur. Il ne faut surtout pas se fier à l'estimation $\hat{c}$ de la taille de la collusion fournie par l'algorithme EM pour accuser un nombre donné de personnes, comme par exemple, accuser les $\hat{c}$ plus gros scores. Il est plus sage de rester sur une attitude plus s˚re qui est d'accuser uniquement le plus grand score, même si les scores sont calculés sur une hypothétique taille de collusion.

La figure 3(d) montre que quand la taille de la collusion est grande, l'algorithme EM joue son rôle avec précision et les performances de notre décodage sont bien meilleures que le décodage de Tardos, mais elles sont très dépendantes de la stratégie de la collusion.

4. Conclusion

Nous avons exhibé de nouvelles fonctions d'accusation, meilleures que les fonctions précédentes lorsque nous identifions correctement la stratégie de la collusion. Le gain de performance est très inégal selon les stratégies. Le décodage originel de G. Tardos et B. Skoric se comporte de la même façon quelle que soit la stratégie, masquant le fait que les stratégies déterministes sont bien moins dangereuses que les probabilistes. La stratégie « Uniforme » est la plus dure à contrer. La question qui reste à résoudre est de trouver pourquoi certaines stratégies de collusion sont plus dures à contrer que d'autres pour une taille de collusion donnée c, et ainsi identifier la pire d'entre elles. La structure itérative de notre décodeur nous permet d'estimer dynamiquement la taille et la stratégie de la collusion. L'estimation de c est un sujet qui devrait être amélioré, bien qu'une mauvaise estimation de c permet parfois d'obtenir de meilleurs résultats. Il faudrait alors optimiser les fonctions quel que soit le nombre de *colluders*.

Bibliographie

Blayer O., Tassa T. (2008). « Improved versions of Tardos'fingerprinting scheme », *Des. Codes cryptography*, vol. 48, p. 79-103.

Boneh D., Shaw J. (1998). « Collusion-secure fingerprinting for digital data », *IEEE Transactions of Information Theory*.

Chor B., Fiat A., Naor M., Pinkas B. (2000). « Tracing traitors », *IEEE Transactions of Information Theory*, vol. 46, p. 893-910.

Delyon B. (2010). « Régression ».

Furon T., Guyader A., Cerou F. (2008a). « Experimental assessment of the reliability for watermarking and fingerprinting schemes », *EURASIP Journal on Information Security*.

Furon T., Guyader A., Cerou F. (2008b). « On the design and optimization of Tardos probabilistic fingerprinting codes », *Information Hiding 2008, Santa Barbara, California, USA*.

Peikert C., Shelat A., Smith A. (2003). « Lower bounds for collusion-secure fingerprinting », p. 472-478.

Schaathun H. (2004). « Binary collusion-secure codes : comparison and improvements », rapport technique, *Université de Bergen, Département d'informatique, Bergen, Norvège*.

Skoric B., Katzenbeisser S., Celik M. (2008). « Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes », *Designs, Codes and Cryptography*, vol. 46, n° 2, p. 137-166, February.

Staddon J., Stinson D., Wei R. (2001). « Combinatorial properties of frameproof and traceability codes », *IEEE Transactions of Information Theory*, vol. 47, p. 1042-1049.

Tardos G. (2003). « Optimal probabilistic fingerprint codes », *Proc. of the 35th annual ACM symposium on theory of computing*, ACM, San Diego, CA, USA, p. 116-125.

Annexes

Détails des calculs

On veut trouver les fonctions $g_{11}(p,\theta)$, $g_{10}(p,\theta)$, $g_{01}(p,\theta)$ et $g_{00}(p,\theta)$ qui maximisent $\tilde{\mu}_{Coll}$ sous les contraintes $\tilde{\mu}_{Inn} = 0$, $\kappa(S_j,S_k) = 0$, et $\tilde{\nu}_{Inn} = 1$. Tout d'abord, calculons tous les éléments.

Statistiques concernant les utilisateurs innocents

On pose $q(p,\theta) = \mathbb{P}(Y = 1|p,\theta)$, ce qui peut s'exprimer en termes de modèle de collusion par :

$$q(p,\theta) = \sum_{\sigma=0}^{c} \mathbb{P}(Y = 1|\Sigma = \sigma) \binom{c}{\sigma} p^{\sigma}(1-p)^{c-\sigma}.$$

On suppose que les scores des innocents sont centrés. Dans ce cas, ils ne sont pas corrélés si :

$$\begin{aligned}\kappa(S_j,S_k) &= \mathbb{E}_p\left[q(p,\theta)\left(pg_{11}(p,\theta)+(1-p)g_{10}(p,\theta)\right)^2\right.\\ &+ \left.(1-q(p,\theta))\left(pg_{01}(p,\theta)+(1-p)g_{00}(p,\theta)\right)^2\right] = 0.\end{aligned}$$

Ce qui donne :

$$pg_{11}(p,\boldsymbol{\theta}) + (1-p)g_{10}(p,\boldsymbol{\theta}) = 0, \quad [14]$$

$$pg_{01}(p,\boldsymbol{\theta}) + (1-p)g_{00}(p,\boldsymbol{\theta}) = 0. \quad [15]$$

Les fonctions g_{10} et g_{01} sont déterminées par les fonctions g_{11} et g_{00}. Donc, il y a seulement deux fonctions à optimiser.

Y a t-il de nouvelles contraintes sur g_{00} et g_{11} données par la relation $\tilde{\mu}_{Inn} = 0$? Par définition, nous avons :

$$\begin{aligned}\tilde{\mu}_{Inn} &= \mathbb{E}_p\left[q(p,\boldsymbol{\theta})\left(pg_{11}(p,\boldsymbol{\theta}) + (1-p)g_{10}(p,\boldsymbol{\theta})\right)\right.\\ &+ \left.(1-q(p,\boldsymbol{\theta}))\left(pg_{01}(p,\boldsymbol{\theta}) + (1-p)g_{00}(p,\boldsymbol{\theta})\right)\right].\end{aligned}$$

Mais, avec les relations [14] et [15], le terme de droite est égal à zéro. Donc nous n'avons pas de nouvelle relation entre g_{00} et g_{11}. La variance a une expression similaire mais avec les fonctions au carré, et peut se simplifier en utilisant (14) et (15) :

$$\begin{aligned}\tilde{v}_{Inn} &= \mathbb{E}_p\left[q(p,\boldsymbol{\theta})\left(pg_{11}^2(p,\boldsymbol{\theta}) + (1-p)g_{10}^2(p,\boldsymbol{\theta})\right)\right.\\ &+ \left.(1-q(p,\boldsymbol{\theta}))\left(pg_{01}^2(p,\boldsymbol{\theta}) + (1-p)g_{00}^2(p,\boldsymbol{\theta})\right)\right]\\ &= \mathbb{E}_p\left[q(p,\boldsymbol{\theta})\frac{p}{1-p}g_{11}^2(p,\boldsymbol{\theta}) + (1-q(p,\boldsymbol{\theta}))\frac{1-p}{p}g_{00}^2(p,\boldsymbol{\theta})\right].\end{aligned}$$

Statistique concernant les colluders

Nous simplifions l'expression de la moyenne des scores des colluders :

$$\begin{aligned}\tilde{\mu}_{Coll} = \mathbb{E}_p \quad [\quad & p\mathbb{P}(Y=0|X=1,p,\boldsymbol{\theta})g_{01}(p,\boldsymbol{\theta})\\ + \quad & p\mathbb{P}(Y=1|X=1,p,\boldsymbol{\theta})g_{11}(p,\boldsymbol{\theta})\\ + \quad & (1-p)\mathbb{P}(Y=0|X=0,p,\boldsymbol{\theta})g_{00}(p,\boldsymbol{\theta})\\ + \quad & (1-p)\mathbb{P}(Y=1|X=0,p,\boldsymbol{\theta})g_{10}(p,\boldsymbol{\theta})].\end{aligned}$$

En utilisant les relations [14] et [15], et les propriétés

$$\begin{aligned}\mathbb{P}(Y=0|X=0,P=p,\boldsymbol{\theta}) + \mathbb{P}(Y=1|X=0,P=p,\boldsymbol{\theta}) &= 1\\ \mathbb{P}(Y=0|X=1,P=p,\boldsymbol{\theta}) + \mathbb{P}(Y=1|X=1,P=p,\boldsymbol{\theta}) &= 1\end{aligned}$$

on obtient :

$$\tilde{\mu}_{Coll} = \mathbb{E}_p\left[A(p,\boldsymbol{\theta})\left(pg_{11}(p,\boldsymbol{\theta}) + (1-p)g_{00}(p,\boldsymbol{\theta})\right)\right], \quad [16]$$

avec $A(p,\boldsymbol{\theta}) = \mathbb{P}(Y=1|X=1,P=p,\boldsymbol{\theta}) - \mathbb{P}(Y=1|X=0,P=p,\boldsymbol{\theta})$. Ces termes sont calculés comme ceci :

$$\mathbb{P}(Y=1|X=1,p,\boldsymbol{\theta}) = \sum_{\sigma=1}^{c} \mathbb{P}(Y=1|\Sigma=\sigma) \begin{pmatrix} c-1 \\ \sigma-1 \end{pmatrix} p^{\sigma-1}(1-p)^{c-\sigma},$$

$$\mathbb{P}(Y=1|X=0,p,\boldsymbol{\theta}) = \sum_{\sigma=0}^{c-1} \mathbb{P}(Y=1|\Sigma=\sigma) \begin{pmatrix} c-1 \\ \sigma \end{pmatrix} p^{\sigma}(1-p)^{c-\sigma-1}.$$

Lagrangien

On veut maximiser $\tilde{\mu}_{Coll}$ sous les contraintes que $\tilde{v}_{Inn} = 1$, en utilisant un Lagrangien :

$$J(g_{11}(p,\boldsymbol{\theta}),g_{00}(p,\boldsymbol{\theta})) = \tilde{\mu}_{Coll} - \lambda(\tilde{v}_{Inn} - 1).$$

Cette expression atteint un extremum lorsque de faibles perturbations des fonctions d'entrée ne changent pas sa valeur. Nous définissons une dérivée en considérant la fonction comme un terme linéaire de l'expansion de Taylor :

$$J(g_{11}(p,\boldsymbol{\theta}) + \varepsilon(p), g_{00}(p,\boldsymbol{\theta})) = J(g_{11}(p,\boldsymbol{\theta}),g_{00}(p,\boldsymbol{\theta})) + \frac{\partial J(g_{11}(p,\boldsymbol{\theta}),g_{00}(p,\boldsymbol{\theta}))}{\partial g_{11}(p,\boldsymbol{\theta})}$$

$$+ \mathbb{E}_p[o(\varepsilon(p))].$$

$$\frac{\partial J(g_{11}(p,\boldsymbol{\theta}),g_{00}(p,\boldsymbol{\theta}))}{\partial g_{11}(p,\boldsymbol{\theta})} = \mathbb{E}_p\left[p\varepsilon(p)\left(A(p,\boldsymbol{\theta}) - 2\lambda\frac{q(p,\boldsymbol{\theta})}{1-p}g_{11}(p,\boldsymbol{\theta})\right)\right].$$

Ceci est égal à zéro pour tout $\varepsilon(p)$ si :

$$g_{11}(p,\boldsymbol{\theta}) = \frac{1}{2\lambda}\frac{1-p}{q(p,\boldsymbol{\theta})}A(p,\boldsymbol{\theta}). \qquad [17]$$

En annulant la dérivée selon la seconde fonction, on obtient de même :

$$g_{00}(p,\boldsymbol{\theta}) = \frac{1}{2\lambda}\frac{p}{1-q(p,\boldsymbol{\theta})}A(p,\boldsymbol{\theta}), \qquad [18]$$

avec λ donné par la contrainte :

$$\lambda = \frac{1}{2}\sqrt{\mathbb{E}_p\left[A^2(p,\boldsymbol{\theta})\frac{p}{q(p,\boldsymbol{\theta})}\frac{1-p}{1-q(p,\boldsymbol{\theta})}\right]}. \qquad [19]$$

On insère [18] et [19] dans [16], ce qui donne :

$$\tilde{\mu}_{Coll} = \sqrt{\mathbb{E}_p\left[A^2(p,\boldsymbol{\theta})\frac{p}{q(p,\boldsymbol{\theta})}\frac{1-p}{1-q(p,\boldsymbol{\theta})}\right]}. \qquad [20]$$

Article reçu le 1/10/2009
Version révisée le 15/05/2010

Ana Charpentier *a obtenu son Master Recherche Mathématiques Cryptologie Codage et Calcul à l'Université de Limoges en 2007. Elle a commencé en 2008 une thèse à l'INRIA Rennes Bretagne Atlantique sur les codes traçants probabilistes sous la supervision de Caroline Fontaine et Teddy Furon.*

Caroline Fontaine *est titulaire du doctorat en informatique de l'université Paris 6, obtenu en 1998. Elle est chargée de recherche au CNRS, affectée depuis octobre 2009 au Lab-STICC (UMR 3192) à Brest. Ses travaux portent sur la protection de l'information, et s'appuient sur la cryptographie, les codes correcteurs d'erreur, la théorie de l'information et le traitement du signal. Co-responsable du parcours « sécurité » du Master 2 Recherche en Informatique de Rennes, elle est aussi membre du comité éditorial de la revue « Journal in Computer Virology ».*

Teddy Furon *a reçu les diplômes de DEA en télécommunications numériques (1998) et de thèse en traitement du signal (2002) de l'École Nationale Supérieure des Télécommunications de Paris. De 1998 à 2001, il a travaillé sur les techniques de tatouage pour la protection de copie en tant qu'ingénieur de recherche au laboratoire Sécurité de Thomson R\&D France à Rennes. Il a continué cette thématique en postdoc au laboratoire TELE de l'Université de Louvain-la Neuve en Belgique. Il est maintenant Chargé de Recherche à l'INRIA Rennes Bretagne Atlantique, expert en protection des contenus multimedia.*