Décodage EM du code de Tardos pour le fingerprinting

Ana Charpentier, Caroline Fontaine, Teddy Furon

Décodage EM du code de Tardos pour le fingerprinting

Ana CHARPENTIER[1], Caroline FONTAINE[2], Teddy FURON[3]

[1]INRIA, Centre Rennes - Bretagne Atlantique
Campus de Beaulieu,35042 Rennes, France

[2]CNRS-IRISA, Centre Rennes - Bretagne Atlantique
Campus de Beaulieu,35042 Rennes, France

[3]Thomson Security Lab
Cesson Sevigne, France

Ana.Charpentier@irisa.fr, Caroline.Fontaine@irisa.fr
Teddy.Furon@thomson.net

Thème – Communication et codage : Cryptage et tatouage (Traçage de documents marqués)

Problème traité – Identifier les utilisateurs malhonnêtes ayant participé à la création d'une copie pirate d'un document numérique.

Originalité – Utilisation d'un algorithme itératif et de nouvelles fonctions pour la partie accusation du code de Tardos.

Résultats – Amélioration de la phase d'accusation qui permet de réduire la taille du code.

1 Contexte

Nous parlons dans cet article de *fingerprinting*, désigné aussi sous les noms de *traitor tracing*, ou *transactional watermarking*. Le problème est le suivant : un distributeur de données numériques (images, audio, vidéo,...) distribue des copies d'un contenu à n utilisateurs. Quelques uns de ces utilisateurs sont malhonnêtes, aussi appelés *colluders*, et mélangent leurs contenus pour créer une copie pirate qui est redistribuée illégalement. On souhaite retrouver leur identité en analysant la copie pirate. Pour cela, on associe une technique de marquage robuste avec un code anti-collusion, partie étudiée ici.

2 Code de Tardos

En 2003, G. Tardos a présenté une famille de codes de fingerprinting probabilistes très efficaces [2]. Leur intérêt réside surtout dans leur taille réduite et dans la facilité de génération des mots de code. B. Skoric *et al* ont proposé un décodage symétrique de ces codes, ainsi qu'une version pour alphabet q-aire [3]. Ces articles utilisent des fonctions d'accusations qui sont les mêmes pour tous les types d'attaques. En 2008, Furon *et al* [1] ont montré que ces fonctions étaient bien les meilleures qu'on pouvait utiliser dans un contexte général, mais qu'elles pouvaient être optimisées si la stratégie des *colluders* est connue.

Commençons par rappeler en quoi consistent les codes de Tardos pour un alphabet binaire [3]. Soit n le nombre d'utilisateurs et m la longueur du code. Soit $\mathbf{X}$ la matrice contenant l'ensemble des mots du code. On désigne par $\mathbf{X}_j = (X_{j1}, X_{j2}, \ldots, X_{jm})$ le mot de code de l'utilisateur j. On tire m valeurs $p_i \in [0,1]$ indépendantes et identiquement distribuées suivant la pdf $f(p) = \frac{1}{\pi\sqrt{p(1-p)}}$. Chaque élément de la matrice est alors tiré de façon aléatoire, en suivant $\mathbb{P}(X_{ji} = 1) = p_i$.
Chaque utilisateur reçoit une copie contenant une marque différente. Pour la partie accusation, soit $\mathbf{Y}$ la marque extraite de la copie pirate. Avec cette marque et le mot de code qui lui est associée, on calcule un score S_j d'accusation pour chaque utilisateur j. Le score est calculé de telle sorte que les plus grands scores sont en espérance ceux des *colluders*. Pour ce calcul, on utilise quatre fonctions d'accusations. $S_j = \sum_{i=1}^{m} U(Y_i, X_{ji}, p_i)$

$$U(1,1,p) = g_{11}(p), \qquad U(0,0,p) = g_{00}(p), \qquad U(0,1,p) = g_{01}(p), \qquad U(1,0,p) = g_{10}(p).$$

Dans le schéma de Tardos symétrique, les fonctions sont les suivantes : $g_{11}(p) = g_{00}(1-p) = -g_{01}(p) = -g_{10}(1-p) = \sqrt{\frac{1-p}{p}}$. Ce sont ces quatre fonctions qui sont modifiées dans notre schéma.

3 Un schéma itératif pour calculer les scores

Nous voulons poursuivre le travail entamé par Furon *et al* [1] en adaptant les fonctions d'accusation à la stratégie utilisée par les pirates, et aller plus loin en estimant cette stratégie. Nous avons pour ce faire construit un schéma itératif utilisant un algorithme EM (Expectation-Maximization). On suppose que la stratégie est la même pour toutes les positions.

1. Initialisation : calcul des scores avec les fonctions de Tardos, on obtient la séquence de n scores $\mathbf{S}$.
2. Décodage EM : la séquence $\mathbf{S}$ est un mélange de scores de *colluders* et d'innocents. Un algorithme EM classique prend la séquence $\mathbf{S}$ en entrée. En sortie, il donne la séquence $\hat{\mathbf{T}}^{(k)}$, dans laquelle $\hat{T}_j^{(k)}$ est la probabilité que le score $S_j^{(k)}$ de l'utilisateur j soit celui d'un *colluder*.
3. Estimateur de densité : avec $\hat{c}^{(k)}$ (le nombre estimé de *colluders*) et la séquence $\hat{\mathbf{T}}^{(k)}$, on estime la stratégie des *colluders* par la probabilité $\hat{\mathbb{P}}^{(k+1)}(Y = 1|\Sigma = \sigma)$.
4. En optimisant la distance de Kullback-Leibler entre les distributions des *colluders* et des innocents, on trouve des fonctions $g_{11}(p)$, $g_{00}(p)$, $g_{01}(p)$ et $g_{10}(p)$ adaptées à la stratégie des *colluders* $\hat{\mathbb{P}}^{(k+1)}(Y = 1|\Sigma = \sigma)$. Ces fonctions nous permettent de calculer de nouveaux scores $\mathbf{S}^{(k+1)}$ et on peut alors recommencer le processus à l'étape 2.

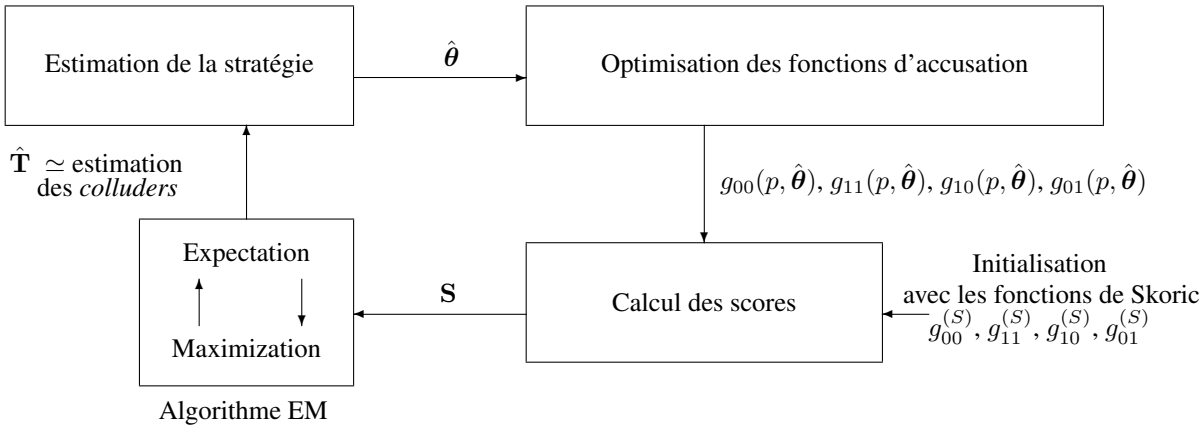

FIG. 1 – Schéma itératif du calcul de score

Les fonctions d'accusation sont calculées avec la contrainte que les scores des utilisateurs innocents sont décorrélés. Pour une longueur de code m assez grande, les scores sont considérés comme suivant une distribution gaussienne. Les scores des innocents sont donc indépendants, c'est une condition nécessaire à l'application de l'algorithme EM. Mais cette hypothèse est fausse en ce qui concerne les scores des *colluders*. Nous avons pu remarquer que ceci n'influe pas sur la convergence de l'algorithme tant que le nombre de *colluders* est faible comparé au nombre total d'utilisateurs ie. $c \ll n$.

4 Résultats

Nous avons considéré le même jeu de stratégies des *colluders* que celles étudiées dans l'article [1], pour comparer nos résultats.
Uniforme : Les *colluders* choisissent au hasard un symbole parmi ceux qu'ils ont sur leurs copies, $\mathbb{P}(Y = 1|\Sigma = \sigma) = \sigma/c$;
Majorité : les *colluders* choisissent le symbole le plus fréquent : $\mathbb{P}(Y = 1|\Sigma = \sigma) = 1$ if $\sigma > c/2$, 0 sinon ;
Minorité : les *colluders* choisissent le symbole le moins fréquent : $\mathbb{P}(Y = 1|\Sigma = \sigma) = 0$ if $\sigma > c/2$, 1 sinon ;
All1 : s'ils ont au moins un '1', les *colluders* mettent un '1' : $\mathbb{P}(Y = 1|\Sigma = \sigma) = 1$ si $\sigma \neq 0$;
All0 : s'ils ont au moins un '0', les *colluders* mettent un '0' : $\mathbb{P}(Y = 1|\Sigma = \sigma) = 0$ si $\sigma \neq c$.

Résultats théoriques. Notre schéma donne de plus grandes distance de Kullback-Leibler entre les deux distributions des scores des innocents et des *colluders* que le décodage de Tardos (pour Tardos, la valeur est la même quelles que soient les stratégies), même si la stratégie estimée est imprécise.

Résultats expérimentaux. Par une simulation Monte Carlo, nous mesurons la probabilité d'accuser à raison l'utilisateur ayant le i-eme plus grand score, et ce pour les stratégie décrites précédemment.

FIG. 2 – Comparaison entre le décodage de Tardos (à droite) et le notre (à gauche) pour les stratégies considérée. Probabilité d'accuser correctement le i-ième plus grand score : $i \in \{ 1,\ldots,8\}$, $m = 1000$, $c = 8$, $n = 5000$, 100 expériences.

Ce graphique montre que nous accusons plus de colluders qu'avec le schéma classique. Le résultat le plus significatif est l'énorme différence des résultats en fonction des stratégies. On constate vraiment que certaines stratégies se combattent beaucoup mieux que d'autres. Les stratégies déterministes sont globalement mieux estimées que les stratégies aléatoires, comme la stratégie uniforme, par exemple.

5 Conclusion

Nous avons trouvé des fonctions d'accusations qui fonctionnent beaucoup mieux que les originales lorsqu'on arrive à estimer la stratégie correctement. Ces fonctions permettent des longueurs de code inférieures à celles requises pour les codes de Tardos (qui sont déjà des codes assez courts). Nous avons aussi construit un schéma itératif estimant cette stratégie, ainsi que la taille de la collusion. En revanche, notre schéma fonctionne mieux lorsque le nombre de *colluders* est élevé. En effet, si c est trop petit, l'algorithme EM donne des résultats imprécis : l'estimation de c, et par la suite de la stratégie, ne se fait plus correctement. Les résultats sont très différents selon la stratégie des *colluders*, ce qui était masqué avec le décodage de Tardos qui est insensible à cette donnée.

Références

[1] T.Furon, A.Guyader et F.Cerou. *On the design and optimization of Tardos probabilistic fingerprinting codes*. Information Hiding, 2008

[2] G.Tardos. *Optimal probabilistic fingerprint codes*. Proc. of the 35th annual ACM symposium on theory of computing, 2003

[3] B.Skoric, S.Katzenbeisser et M.Celik. *Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes*. Designs, Codes and Cryptography, 2008.