Le traçage de traîtres

Teddy Furon

Le traçage de traîtres

Teddy Furon

Thomson Security Lab, Cesson-Sévigné, France

Résumé Ce papier présente de façon très simple le traçage de traîtres redistribuant illégalement des contenus multimedia. Son originalité réside dans le fait que les solutions (modélisations, idées de base, contributions) venant de trois communautés très différentes y sont détaillées : les cryptographes, les statisticiens et les traiteurs de signaux.

1 Introduction

Le traçage de traîtres est aussi connu sous les noms anglais de *transactional watermarking*, *content serialisation*, *users forensics* ou encore *active fingerprinting*. Gare à la confusion, *fingerprinting* a aussi beaucoup d'autres sens. Ici, la métaphore est qu'un utilisateur en consommant un contenu multimedia touche celui-ci et donc laisse ses empreintes qui serviront à l'identifier. L'application type est la vidéo à la demande sur Internet. Un serveur distribue des copies personnalisées d'un même contenu à n utilisateurs. Parmi ceux-ci, certains sont malhonnêtes et redistribuent illégalement des copies pirates. Les ayant droits souhaitent connaître l'identité de ces 'sources'. Pour ce faire, un identifiant unique sous la forme d'une séquence de m bits est caché dans chaque vidéo à l'aide d'une technique de tatouage numérique. Ainsi sont produites n copies du contenu, toutes différentes mais pourtant perceptiblement identiques. Cet identifiant permet de 'tracer la source' des copies pirates. Cependant, il se peut que les pirates soient plusieurs, et qu'ils mélangent leurs copies pour brouiller les pistes.

1.1 Le traçage de traîtres est une réalité

Le traçage de traître existe depuis longtemps. Il y a plus d'un siècle, le coeur de métier de certaines sociétés était de vendre des tables de valeurs mathématiques ou physiques de 'grande précision', telles que des tables de logarithmes. Les arrondis, supérieur ou inférieur, des dernières décimales encodaient le nom du client qui avait acheté la table. Dans les années 80, irritée par les fuites de documents dans son gouvernement, sport national outre-manche, Margaret Thatcher distribua aux membres de son cabinet des documents avec des longueurs d'espace variables. Comme la presse anglaise reproduisaient un fac-simile des documents donnés par le traître, son identité fut vite dévoilée.

Concernant l'industrie du divertissement, un des premiers déploiements ambitieux de traçage de traîtres a été commis par la défunte entreprise américaine DivX-Corporation (rien à voir avec le format de compression de vidéo) qui en 1996 vendait des lecteurs de DVD améliorés supportant la location. Hélas, un

serveur distant centralisait les données sensibles sur la vie privée des utilisateurs ("Qui regarde quoi ?"), ce qui a entraîné une levée de boucliers aux Etats-Unis et la mort de cette société.

Plus récemment, un tatouage numérique personnalise les screeners. Les screeners sont des DVD de films récents voire pas encore sortis en salle, envoyés aux critiques professionnels ou aux membres de jurys tel celui de la cérémonie des Oscars. En 2004, après enquête du FBI, la détection d'un tatouage a permis de condamner l'acteur Carmine Caridi, connu pour son rôle dans *Le Parrain*, à une amende de 600 000$. Celui-ci a été exclu du jury des Oscars (une première historique), et a passé une période de 4 ans sans tourner le moindre film. Son complice, Sprague, qui a techniquement copié le contenu d'une soixante de screeners de Caridi pour les poster sur un réseau P2P a été condamné à 3 ans de prison où il est mort d'une crise cardiaque.

Aujourd'hui, iTunes Store écrit des données personnelles ainsi qu'une signature numérique dans l'entête des fichiers vendus non-DRMisés. Dans le standard de protection AACS des disques Blu-Ray, il existe un premier tatouage vidéo permettant d'identifier le modèle d'appareil ayant servi pour une fuite de très haute qualité. De plus, le système *SequenceKey* inventé par IBM utilise un véritable code anti-collusion moderne [1]. Des scènes du film sont disponibles en plusieurs versions équivalentes (tatouées différemment). Un lecteur BR n'a la clé de déchiffrement que d'une seule de ces versions à chaque fois. Ainsi, cette séquence de scènes tatouées cache en fait des informations sur l'identifiant de l'appareil. Associé à un schéma de *broadcast encryption*, il est ainsi possible de repérer puis 'tuer' (*blacklisting*) un lecteur cracké.

1.2 Le traçage de traîtres, un DRM 2.0 ?

Outre ces quelques exemples, le traçage de traîtres, avec des technologies comme la reconnaissance de contenu ou le filtrage, est semble-t-il à la mode. Les mesures techniques de protection, autrement dit les *DRM*, ont créé une énorme frustration chez les utilisateurs car elles sont propriétaires, non-interopérables, et trop intrusives. Le futur du DRM est de se retirer hors du réseau domestique, de disparaître des appareils d'électronique grand public, pour se renforcer sur le réseau Internet. Un des moyens est de nettoyer Internet des contenus pirates : interdire l'upload de vidéo copyrightée sur les sites UGC (*User Generated Content*, tel YouTube, DailyMotion) ou filtrer les sites de Peer2Peer grâce à la reconnaissance de contenu. C'est le principe de la fameuse 'réponse graduée' contenue dans la loi HADOPI. Cependant, ce grand nettoyage est à refaire périodiquement, le filtrage à procéder continuellement. Le traçage de traîtres, quant à lui, vise à éradiquer la source de la fuite une bonne fois pour toute.

Deux cas de figures sont à discerner. La source est un professionnel : par exemple, un critique de cinema ou un sous-traitant en post-production. Le traçage de traîtres est un élément de preuve devant un juge et la jurisprudence montre que les traîtres sont pendus haut et court. La source est un particulier : un client d'un service VoD ou un abonné à un bouquet de chaînes numériques qui 'streame' un flux DVB sur Internet. Le traçage de traîtres sert juste à identifier

l'utilisateur dont l'abonnement ou le contrat de vente sera suspendu. On imagine une liste commune à tous les fournisseurs de contenus d'utilisateurs 'blacklistés'.

2 Le traçage de traîtres à la mode crypto

2.1 L'approche historique

C'est la communauté cryptographique qui a été la première à étudier le sujet. Ce dernier est très similaire à un problème de gestion de clés secrètes familier à cette communauté. Imaginons que les utilisateurs malhonnêtes cassent leurs décodeurs pour trouver la clé secrète, et créent des décodeurs pirates. Si tous les utilisateurs ont une clé propre, il est facile à partir d'un décodeur pirate de retrouver l'identité des traîtres, en revanche le contenu adressé aux abonnés est chiffré et transmis n fois. A l'inverse, si tous les utilisateurs partagent la même clé, le contenu est chiffré et transmis qu'une seule fois, mais il est impossible d'identifier les traîtres. Chor, Fiat et Naor proposent des schémas attribuant un jeu de clés propre à chaque décodeur qui minimise le nombre de chiffrement / transmission et qui permet de retrouver les traîtres même si ceux-ci ont mélangé leurs jeux de clés dans le décodeur pirate [2]. De nos jours, avec des bandes passantes de plus en plus grandes et de moins en moins chères, la mode chez les pirates est de décrypter le contenu et de le retransmettre plutôt que de fabriquer des décodeurs pirates. D'où un saut du traçage de décodeurs au traçage de contenus.

L'article [2] introduit un concept capital : la collusion. La collusion est un vieux mot français définissant une entente secrète entre plusieurs personnes pour nuire à un tiers. Il y a $c \geq 1$ pirates qui mélangent leurs contenus (ou leurs jeux de clés pour [2]) pour déjouer le système de traçage. Ce concept n'a pour l'instant aucune réalité. A ma connaissance, aucune copie pirate n'a impliqué un mélange de contenus en vue de se soustraire à un système de traçage. La collusion ne serait qu'une chimère académique, un délire de chercheurs. En fait, elle est surtout une arme de défense pour l'accusé. Bien qu'ayant agi seul, le pirate identifié peut prétendre être la victime d'une collusion : si on retrouve son identifiant dans la copie pirate, c'est à cause d'un mélange fait de manière intentionnelle par d'autres utilisateurs malhonnêtes. Le coupable se présente en victime et la preuve n'est plus recevable. Le système doit résister à la collusion pour contrer cet argument.

Un modèle mathématique. Boneh et Shaw [3] sont les premiers à faire le lien avec l'article de Chor, Fiat et Naor. Ils définissent un modèle mathématique de la collusion connu sous le nom de *marking assumption*.

- Un contenu est une suite de symboles. Ici, on envisage un alphabet binaire.
- Dans ce contenu, il y a m emplacements peu importants, au sens où on peut y modifier le symbole sans dégrader significativement le contenu. Le mot de code, une séquence binaire de longueur m identifiant un utilisateur, sera caché dans le contenu à ces emplacements.

- Les pirates ne connaissent pas a priori ces emplacements. C'est en comparant leurs copies qu'ils distinguent des différences dévoilant certains emplacements.
- Ils créent une copie en mélangeant symbole par symbole leurs copies.
- Le processus d'accusation connaît ces emplacements et extrait de la copie pirate une séquence, dite séquence pirate, de m symboles.

Ainsi, la règle d'or de ce modèle est que là où les mots de code des pirates ont tous le même symbole, ce dernier se retrouve forcément dans la séquence pirate.

Le mot clé est laché : code. Un code $\mathcal{X}$ est un ensemble de n séquences (ou mots de code) composées de m symboles. Ici les symboles sont binaires, sauf mention contraire. Parmi les 2^m séquences possibles, le code n'en retient que n. La collusion $\mathcal{C}$ est l'ensemble des c mots de code des pirates. Une notion utile est l'ensemble des descendants, $\text{desc}(\mathcal{C})$ qui est l'ensemble de toutes les séquences pirates réalisables à partir des c mots de code en suivant les règles de la *marking assumption*. En binaire, cet ensemble est de taille $2^{m'}$, où m' est le nombre d'emplacements dans les mots de code des pirates où leurs c symboles ne sont pas tous égaux.

Une terminologie des codes anti-collusion. Une terminologie des codes anti-collusion s'est rapidement mis en place [4]. Elle classe les codes en 4 catégories suivant certaines propriétés : *frameproof*, *secure frameproof*, *identifiable parents property* (IPP), et *traceable*. Les deux premières propriétés seront détaillées. A chaque fois, un code a une propriété pour une nombre donné de pirates (ou *colluders* en anglais) : un code est par exemple c-frameproof.

Frameproof. To frame en anglais veut dire produire des fausses preuves pour qu'un innocent soit accusé à tort. Un code est *c-frameproof* s'il est impossible pour une collusion de taille au plus c de recréer le mot de code d'un innocent :

$$\text{desc}(\mathcal{C}) \bigcap \mathcal{X} = \mathcal{C}.$$

Si les pirates ne sont pas idiots, la séquence pirate ne sera donc pas un mot de code (ni un des leurs, ni celui d'un innocent). Cette propriété est un strict minimum et ne dit rien sur la façon de retrouver un traître.

Secure frameproof. Un code est *c-secure frameproof* si aucune collusion de taille c ne peut créer une séquence pirate qu'un autre groupe de c personnes aurait pu créer :

$$\mathbf{y} \in \text{desc}(\mathcal{C}) \bigcap \text{desc}(\mathcal{C}') \Rightarrow \mathcal{C} \bigcap \mathcal{C}' \neq \emptyset.$$

On voit un peu mieux comment peut fonctionner l'accusation. Si la séquence pirate $\mathbf{y}$ appartient à deux ensembles de descendance $\text{desc}(\mathcal{C})$ et $\text{desc}(\mathcal{C}')$, on ne peut décider laquelle des collusions la produit. Cependant, cette propriété nous assure que les deux collusions $\mathcal{C}$ et $\mathcal{C}'$ ont une intersection non vide, et, sans se tromper, on accuse le ou les utilisateurs communs. Mais il reste des problèmes. Imaginons que finalement $\mathbf{y} \in \text{desc}(\mathcal{C}) \bigcap \text{desc}(\mathcal{C}') \bigcap \text{desc}(\mathcal{C}'')$ alors

que $\mathcal{C} \bigcap \mathcal{C}' \bigcap \mathcal{C}'' = \emptyset$ (une configuration en triangle comme sur la figure 1). Qui accuse-t-on ? Un utilisateur de $\mathcal{C} \bigcap \mathcal{C}'$, de $\mathcal{C} \bigcap \mathcal{C}''$ ou de $\mathcal{C}' \bigcap \mathcal{C}''$?

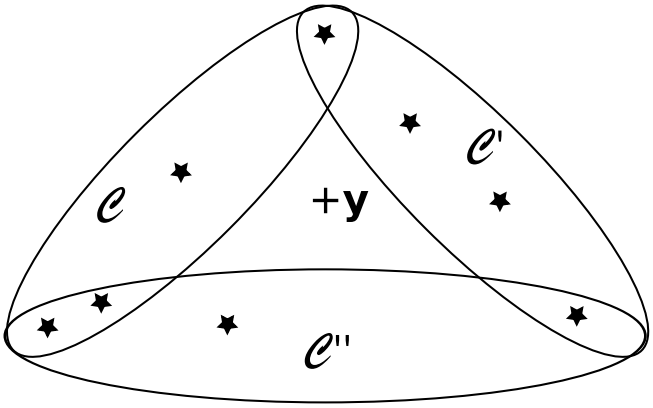

Fig. 1. La séquence pirate $\mathbf{y}$ appartient aux ensembles de descendance de trois collusions $\mathcal{C}$, $\mathcal{C}'$ et $\mathcal{C}''$. Les mots de code sont figurés sous forme d'étoiles. Qui accusez-vous ?

Il faut une propriété encore plus remarquable, la traçabilité forte (donnée par les codes *IPP* ou *traceable*), pour assurer une accusation rigoureuse. Le fin du fin en traçabilité forte est la propriété dite, en anglais, *traceable* où l'accusation se déduit de la définition : un code est *c-traceable* si le mot de code le plus proche au sens de Hamming de la séquence pirate est celui d'un traître. Cependant la traçabilité forte est une propriété très contraignante : par exemple, il est impossible de faire un tel code à partir d'un alphabet binaire si $c > 2$. En général, il faut de grand alphabet q-aire ($q > c^2$) et de très très longues séquences.

Au contraire, la traçabilité faible propose un changement complet de stratégie en admettant que l'accusation peut donner des erreurs. Cette relaxation des contraintes donne des longueurs de code plus courtes comme on le verra dans la Section 3.

L'outil de base : le code correcteur d'erreurs. Le théorème le plus connu en traçabilité forte fait le lien entre le traçage de traîtres et la théorie des codes correcteurs d'erreurs [5] : si $\mathcal{X}$ est un code correcteur d'erreurs (n mots de code de longueur m), de distance minimale $d > m(1 - c^{-2})$ alors $\mathcal{X}$ est un code c-traceable.

Pour faire court, considérons le mot de code $\mathbf{x}$ d'un traître donné. En mélangeant leurs mots de code, ils forment une séquence pirate $\mathbf{y}$ qui peut être vue comme $\mathbf{x} + \mathbf{e}$, c'est à dire le mot de code du traître plus des erreurs. L'algorithme de décodage du code correcteur d'erreur enlèvera les erreurs et retrouvera

le mot de code $\mathbf{x}$ du traître. Si les traîtres partagent les risques, un symbole sur c provient de $\mathbf{x}$, et il y a au plus $m(1-c^{-1})$ erreurs. La condition sur la distance minimale assure qu'aucun innocent est plus proche de $\mathbf{y}$. Cependant, décoder autant d'erreurs n'est pas à la portée de tous les codes correcteurs. Il faut employer des codes extrêmement redondants ou des concaténation de codes produisant des mots de code très très longs. Si le théorème ci-dessus stipule que des codes correcteurs d'erreurs sont utiles en traçage de traîtres, l'analogie avec un canal de transmission bruité n'est très convaincante et indique que ce n'est pas vraiment le meilleur outil.

3 Le traçage de traîtres à la mode statistique

Devant les longueurs énormes des codes à traçabilité forte, les cryptographes ont relâché les contraintes et toléré des erreurs d'accusation. C'est la traçabilité faible. Il y a deux types d'erreurs :

- La probabilité ϵ_1 d'accuser à tort des innocents,
- La probabilité ϵ_2 de rater des pirates.

Le code est utile si on sait borner ces erreurs, et si les bornes sont très faibles. ϵ_1 est la probabilité la plus critique, typiquement de l'ordre de 10^{-6}. ϵ_2 est beaucoup plus grande, de l'ordre de 10^{-1} car on tolère que de temps en temps les traîtres nous échappent.

Pour comparer deux codes, l'habitude est de travailler à une taille de collusion donnée c, d'imposer les probabilités ϵ_1 et ϵ_2, et de comparer les longueurs m des mots de code nécessaires pour atteindre ce niveau de performance. Le meilleur code a la plus petite longueur.

Tardos le génie inconnu. L'apport des statisticiens est tout naturel : il faut savoir estimer et surtout borner des probabilités, c'est leur métier. Les premières bornes proposées par les cryptographes ont été raffinées, ce qui donne des titres d'articles comme, par exemple, "*The Boneh-Shaw fingerprinting scheme is better than we thought*" [6].

Là où c'est plus étonnant, c'est que l'un d'entre eux a proposé le code le plus efficace connu jusqu'à maintenant en traçabilité faible. Gabor Tardos est un maître en probabilité, statistiques, et calculs combinatoires. Il est connu pour avoir démontré avec Adam Marcus la conjecture de Stanley-Wilf. En 2003, un de ses collègues lui explose la problématique de traçage de traîtres. Notamment, un résultat non-construcutif très récent de la communauté cryptographique [7] : la borne inférieure la plus fine sur la longueur d'un code binaire est en $O(c^2 \log \epsilon_1 n^{-1})$, mais on ne connaît pas de code atteignant cette borne. Tardos est le premier a exhibé un tel code, de plus il est d'une simplicité déconcertante (une dizaine de lignes en Matlab). Mais, il publie dans une conférence qui est inconnue des tatoueurs et des cryptographes [8], et du coup ses résultats restent dans l'ombre pendant 2 ans. Ils sont remis à la mode par l'équipe de Philips [9]. Autre fait remarquable : Tardos ne dit rien sur son raisonnement. Il

donne la construction du code et il montre qu'il atteint la borne. Pourquoi une telle construction ? pourquoi ces valeurs de paramètres ? Mystère.

Un nouveau modèle de collusion. Les cryptographes ont inventé la notion d'ensemble de descendance, qui est une liste finie de toutes les séquences pirates possibles à partir des c mots de code des *colluders*. Cependant, pour calculer la probabilité d'accuser un innocent, on doit passer en revue toutes les séquences de cet ensemble et à chaque fois voir si un innocent est accusé. Ainsi, $\epsilon_1 = n.n_{accusation}/2^{m'}$. Mais ceci implique qu'une séquence pirate est aussi probable qu'une autre. Ce n'est pas forcément le cas. Les statisticiens préfèrent un modèle statistique (surprenant, non ?) de collusion : $\Pr_Y[1|\Sigma = \sigma]$, $\forall\sigma \in \{0, 1, \ldots, c\}$. La collusion est définie par $c+1$ valeurs qui indique la probabilité que les colluders collent le symbole '1' aux emplacements où ils ont σ fois ce symbole dans leurs mots de code. Si la collusion suit la *marking assumption*, alors $\Pr_Y[1|\Sigma = 0] = 0$ et $\Pr_Y[1|\Sigma = c] = 1$. Avec ceci, on peut modéliser :

- un tirage aléatoire : les colluders lancent un dé à c faces pour savoir de quel mot de code est issu le prochain symbole de la séquence pirate : $\Pr_Y[1|\Sigma = \sigma] = \sigma/c$.
- un vote majoritaire : les colluders mettent le symbole qu'ils ont le plus : $\Pr_Y[1|\Sigma = \sigma] = 1$ si $\sigma > c/2$ (le contraire pour un vote minoritaire).
- un pile ou face : les colluders lancent une pièce pour savoir s'ils mettent un '1' ou un '0' pour les emplacements où ils ont le choix : $\Pr_Y[1|\Sigma = \sigma] = 1/2$ si $0 < \sigma < c$.
- Tout à '1' : les colluders collent un '1' dès qu'ils le peuvent : $\Pr_Y[1|\Sigma = \sigma] = 1$ pour $\sigma > 0$ ($\Pr_Y[1|\Sigma = \sigma] = 0$ avec $\sigma < c$ pour Tout à '0').

Il y a en fait une infinité de collusion possible, et uniquement le pile ou face rend la séquence pirate uniformément distribuée sur l'ensemble de descendance. Ce modèle est donc plus adapté à la traçabilité faible.

L'idée de base : le faisceau de preuve. Encore une fois, Tardos n'a jamais donné son raisonnement, donc ce qui est écrit ici est une tentative de vulgarisation très personnelle. Dans les romans d'aventure de votre jeunesse ou les romans policiers, le coup du traître est un grand classique. Le héros ne cesse de tomber dans des embuscades, il se dit qu'il y a un traître parmi ses proches. Il le démasque car seul celui-ci était au courant de tous ses faits et gestes, ie. que tel jour le héros empruntait tel chemin etc. Une autre analogie est le jeu préféré de ma fille "*Qui est-ce ?* ". On pioche une carte représentant un personnage, et l'adversaire tente de trouver son identité en posant des questions : Est-ce une femme ? A-t-elle des lunettes ? etc.

Ainsi, on peut construire un tableau de n lignes (nombre de suspects) et m colonnes (nombre de réponses aux questions posées). On colle un '1' dans la j-ème ligne et la i-ème colonne si ce suspect répond à ce critère. Cependant, on est sûr d'identifier le personnage au bout d'un nombre fini m d'informations (à moins de poser des questions stupides, c'est à dire non-discrimiantes) : c'est le seul qui voit toutes ses cases remplies de '1'. En traçage de traîtres, tout est plus

difficile à cause de la collusion : essayez de jouer à "*Qui est-ce ?* " avec c cartes. Aux questions de votre adversaire, vous tirez secrètement une des c cartes pour savoir lequel des c personnages servira à faire la réponse.

L'idée est de conserver un tel tableau en traçage de traîtres : si l'utilisateur j a le même symbole que celui présent dans la séquence pirate à l'emplacement i, alors la case (j, i) du tableau reçoit un '1', sinon un '-1'. Avoir le même symbole tend à accuser l'utilisateur, un symbole différent tend à l'innocenter. Cependant, l'accusation ne peut se faire sur un unique symbole, donc on somme la valeur des cases par ligne pour faire un score par utilisateur. C'est l'idée de base de Tardos : le faisceau de preuve. Pour faire vite : un innocent aura un symbole en commun avec la séquence pirate une fois sur deux, d'où statistiquement, un score proche de 0, alors qu'un coupable a en moyenne au moins m/c de ses symboles qui ont servi à faire la séquence pirate, d'où un score en m/c. Bref, les coupables ont statistiquement des scores plus grands que les innocents.

Hélas, les choses ne sont pas aussi simples. Lors de la collusion, m/c symboles proviennent du mot de code d'un coupable, mais quid des autres. Il n'est pas certain que, sur ces $m(1 - c^{-1})$ emplacements restants, les symboles de ce coupable soient décorrélés de ce que ses acolytes y ont mis. Regardons un cas simple : $c = 3$ et la collusion est un vote minoritaire. Le tableau 3 montre que chaque *colluder* a la moitié de ses symboles communs avec la séquence pirate, donc il aura un score proche de zéro comme les innocents.

mot de code 1	0 0 0 0 1 1 1 1
mot de code 2	0 0 1 1 0 0 1 1
mot de code 3	0 1 0 1 0 1 0 1
séquence pirate	0 1 1 0 1 0 0 1

Tab. 1. Vote minoritaire à 3 colluders.

Pour s'en sortir, on favorise certaines situations. Par exemple, quand les *colluders* ont tous le même symbole, celui-ci se retrouve dans la séquence pirate (cf. *marking assumption*) ce qui incrémente leur score d'un point. Si cette situation arrive plus fréquemment sur toute la longueur du code, le score des *colluders* n'est plus nul en moyenne. En revanche, cela ne change rien pour les innocents.

Pour ce faire, dans le code de Tardos, à certains emplacements, beaucoup de mots de code ont le même symbole, disons '1'. Du coup, si ce symbole se retrouve dans la séquence pirate, beaucoup d'utilisateurs voient leur score incrémenté. A l'inverse, peu d'utilisateurs voient leur score incrémenté lorsque, à un emplacement où beaucoup de mots de code ont un '0', il y a pourtant un '1' dans la séquence pirate. Les deux situations sont différentes : dans le premier cas, l'élément de preuve est faible, dans le second il est très fort : les *colluders* ont été assez stupides pour coller un symbole rare (pour leur défense, ils ne le savaient pas). On va donc pondérer l'importance des preuves élémentaires dans le faisceau. Avoir le même symbole augmente le score d'un poids d'autant plus grand

que peu d'utilisateurs sont dans une telle situation. A l'inverse, avoir un symbole différent décroît le score d'un poids d'autant plus grand (en valeur absolue) que peu d'utilisateurs sont dans une telle situation.

Le code de Tardos. Il y a trois phases : initialisation, construction du code, et accusation.

Initialisation : Pour un code de longueur m, tirer au hasard et de manière indépendante m valeurs $\{p(i)\}_{1\leq i\leq m}$ suivant une distribution $f(p) = 1/\pi\sqrt{p(1-p)}$ pour $p \in [0,1]$. Chaque $p(i)$ est compris entre 0 et 1 (en pratique entre t et $1-t$, avec $t \approx 10^{-3}$), et comme $f(p)$ a de fortes valeurs sur les bords, il y a beaucoup de $p(i)$ proches de 0 ou de 1. Ceci se fait en 3 lignes de Matlab :

```
t = 10^-4 ;
tt = asin(sqrt(t)) ;
p = sin(tt + (pi/2-2*tt)*rand(m,1)).^ 2 ;
```

Construction : Pour n utilisateurs, on construit une matrice $n \times m$ en tirant au hasard ses entrées binaires tel que $\Pr_{x(j,i)}[\text{`1'}] = p(i)$. Autrement dit, si $p(i)$ est proche de 1 (resp. 0), beaucoup de mots de code ont un '1' à cet emplacement (resp. un '0'). Soit une seule ligne de Matlab :

```
x = (rand(m,n)<(p*ones(1,n))) ;
```

Accusation : On a extrait la séquence $\mathbf{y}$ de la copie pirate. Le score associé à l'utilisateur j est le suivant : $S_j = \sum_{i=1}^{m} U(y(i), x(j,i), p(i))$ avec :

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

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

Soit 3 lignes de Matlab :

```
Py = ((y==0).*(1-p)+(y==1).*p)*ones(1,c) ;
Smat = (x==y*ones(1,c)).*sqrt((1-Py)./Py)...
       -(x =y*ones(1,c)).*sqrt(Py./(1-Py)) ;
s = sum(Smat,1) ;
```

Les propriétés du code. Si m est relativement grand, le score d'un innocent est distribué comme une Gaussienne centrée sur 0 de variance m alors que le score d'un coupable est distribué comme une Gaussienne centrée sur $2m/(c\pi)$ de variance $\approx m$. On voit bien que plus m est grand ou plus c est petit, plus les deux distributions sont éloignées : en pratique on pourra mieux distinguer les colluders des innocents. Un point fondamental est la garantie que ceci soit vrai pour toutes les stratégie de collusion $\Pr_Y[1|\Sigma = \sigma]$. Il y a pas de stratégie pire (pour l'accusation) que d'autres [10].

La création du code est souple. Si un nouvel utilisateur arrive, il est très simple de générer un nouveau mot de code. De même, la longueur du code n'a pas de valeur stricte. Elle doit cependant être de l'ordre de $m \approx 20c^2 \log(n/\epsilon_1)$.

L'accusation est souple. Une fois les scores calculés, on n'accuse celui qui a le plus gros score si on veut minimiser le risque d'accuser un innocent, ou alors, pour accuser plus d'un *colluder*, on calcule un seuil τ dépendant de ϵ_1 (on travaille à risque constant) tel que l'utilisateur sera accusé si son score est supérieur à τ.

Depuis récemment, on sait convertir un score d'utilisateur en une probabilité que celui-ci soit innocent [11], ce qui nettement plus parlant pour le commun des mortels, juge inclus. Les scores les plus élevés ont bien sûr les probabilités les plus faibles.

L'inconvénient majeur du code de Tardos est son accusation exhaustive. Il est facile de savoir si untel est coupable. En revanche, pour savoir qui sont les coupables, il faut calculer tous les scores, soit une accusation en $O(n)$. Pour un service de VoD, où on sait qui a acheté ce contenu et n varie de l'ordre de mille à un million, l'accusation qui n'a pas de contrainte en temps d'éxécution, est tout à fait réalisable. En revanche, pour l'application DVD, on ne sait pas quel appareil a joué quel contenu. Il faut calculer le scores de 2 milliards d'appareils à chaque fois, ce qui a un coût prohibitif.

4 Le traçage de traîtres à la mode tatouage numérique

Les tatoueurs appartiennent à la communauté traitement du signal. Modéliser un contenu multimedia (une vidéo, un clip audio etc) par une séquence binaire où certains bit seront flippés, leur semble tout simplement un idée incongrue. Autrement dit, la modélisation du problème par les cryptographes (la fameuse *marking assumption* de la section 2.1) est a priori une hérésie. Il y a deux écoles : ceux qui repartent de zéro à partir d'autres modèles mathématiques plus adaptés au tatouage, et ceux qui choisissent une technique de tatouage telle que la *marking assumption* a quand même un sens. On n'évoquera que la seconde école.

Ils faut aussi parler des contraintes liées à l'application. Pour le scénario VoD, la complexité du serveur est ridicule car il sert des milliers de fichiers en même temps. Tatouer à la volée est par conséquent interdit. L'astuce consiste à découper le contenu en bloc, par exemple une seconde de vidéo, et de tatouer tous ses blocs à l'avance autant de fois qu'il y a de symboles : en binaire, on a donc 2 versions d'un bloc : l'un cachant un '1', l'autre un '0'. Ainsi, le catalogue du serveur est deux fois plus gros : tous les contenus existent en double, avec un '1' ou un '0' dans chacun de leurs blocs. Le serveur n'est plus qu'un aiguillage. Le mot de code est caché séquentiellement : suivant le prochain symbole du mot de code de l'utilisateur, il envoie l'une des deux versions du bloc suivant. Le même principe préside pour les DVD Blu-Ray. Le lecteur n'a pas la puissance de calcul pour tatouer en temps réel, par conséquent, certaines scènes du film sont

en plusieurs exemplaires cachant différents symboles, et le lecteur n'en décode qu'une version à la fois.

En multimedia, il existe 3 types d'attaques : les échanges de blocs, les fusions de blocs, et les post-traitements. Le premier type est la collusion étudiée depuis le début de ce papier. Le dernier rassemble des traitements classiques comme le filtrage, l'égalisation, la compression, etc. La technique de tatouage utilisée doit être robuste à ces traitements. Notons que si e symboles sont ainsi effacés, le code de Tardos fonctionne toujours mais sur une longueur $m - e$, ce qu'il gère parfaitement si e est petit devant m. En pratique, tout étant affaire de compromis en tatouage, on étend la taille de blocs, et par conséquent on diminue le débit des symboles cachés, jusqu'à trouver une très bonne robustesse face à ce type d'attaques.

En revanche, le deuxième type est nettement plus critique. Une fusion de blocs consiste par exemple à construire le bloc pirate en moyennant pixel à pixel les blocs tatoués. Qu'est ce qui se passe lorsque l'on fusionne des blocs cachant différents symboles ?

- On retrouve au moins un des symboles. La fusion s'apparente alors à un échange de bloc.
- On ne retrouve aucun symbole. La fusion s'apparente à un effacement comme pour les post-traitements. Sauf qu'ici, les effacements sont systématiques : à chaque fusion de différents blocs, on perd toute information. Sur un code de Tardos, 2 colluders ont des symboles différents sur 13% des blocs, 4 colluders 45%. Autrement dit, e n'est plus petit devant m.
- On retrouve un autre symbole que ceux présents dans les blocs fusionnés (pour un code q-aire, $q > 2$). Ce n'est plus un effacement, mais une erreur au décodage du tatouage qui viole le principe de la *marking assumption*. Les colluders créent un symbole qu'ils n'avaient pas à cet emplacement. L'impact sur le code de Tardos est dramatique : pour un niveau de performance donnée, 5% d'erreurs de décodage (resp. 10%) rallonge le code de 20% (resp. 50%).

Le choix de la technique de tatouage est par conséquent extrêmement délicat.

Une stratégie qui fonctionne bien est la modulation Tout ou Rien (*On-Off Keying* en anglais) [12]. Le tatouage utilisé est une technique dite *zero-bit* : elle cache la présence d'une marque (qui ne code pas un symbole), la détection dit si la marque est présente ou absente dans un bloc, mais il n'y a pas décodage d'un symbole. Cette technique est détournée de sa fonction en utilisant q clés secrètes différentes. Pour cacher un symbole s, on prend la s-ème clé. Utiliser des clés différentes produit des marques indépendantes telles qu'il est impossible en mélangeant certaines d'en imiter une autre. Encore mieux, parfois en fusionnant plusieurs blocs, la détection retrouve la présence de plus d'une marque. Une petite modification de l'accusation de Tardos tire profit de ce supplément d'information. Il s'avère que l'accusation est encore plus fiable. Ainsi, les *colluders* sont prévenus : la fusion de blocs aide plus l'accusation qu'elle ne la perturbe. Du coup, retour à la case départ, les *colluders* sont réduits à l'échange de blocs.

Finalement, tout se passe comme si la *marking assumption* était valide. En binaire, les *colluders* ne peuvent pas transformer un '1' en '0', non parce qu'ils n'ont pas découvert cet emplacement dans la séquence binaire, mais parce qu'ils ne connaissent pas la clé associée à '0' pour tatouer ce bloc.

5 Conclusion

Ce papier montre que la problématique du traçage de traîtres est maintenant bien comprise. On sait quelles sont les bornes théoriques, on connaît des codes performants, des implémentations pratiques voient le jour. Il ne manque plus à vrai dire que la confrontation avec de vraies collusions.

Ce papier passe sous silence d'autres aspects moins classiques de ce problème. Brièvement :

- **Asymétrique** : Schéma où seul l'utilisateur dispose de sa copie personnalisée. Autrement dit, même le serveur ne connaît pas ce contenu. Ainsi, un serveur malhonnête ne peut accuser un innocent en postant sa copie sur un réseau Peer2Peer.
- **Anonyme** : L'utilisateur décline son identité à un tiers de confiance, puis achète de manière anonyme des contenus. Le serveur ayant acquis la preuve qu'un traître existe s'adresse au tiers de confiance pour révéler son identité. Les honnêtes gens restent anonymes.
- **Séquentiel** : Le serveur observe en temps réel la copie pirate. Autrement dit, il existe une boucle de rétro-action entre la copie pirate et le serveur. Ainsi, le prochain symbole caché dans le contenu dépend des précédents décodés dans la copie pirate. C'est typiquement le cas d'un streaming d'une émission d'une chaîne privée.

Références

1. Jin, H., Lotspiech, J., Nusser, S. : Traitor tracing for prerecorded and recordable media. In : Proc. of the 4th ACM workshop on Digital rights management, ACM (2004) 83–90
2. Chor, B., Fiat, A., Naor, M. : Tracing traitors. In Springer-Verlag, ed. : Proc. of Advances in cryptology, CRYPTO'94,. Volume 839., Springer-Verlag (1994) 257–270
3. Boneh, D., Shaw, J. : Collusion-secure fingerprinting for digital data. IEEE Trans. Inform. Theory **44** (1998) 1897–1905
4. Stinson, D.R., Wei, R. : Combinatorial properties and construction of traceability schemes and frameproof codes. SIAM Journal on Discrete Mathematics **11** (1998) 41–53
5. Chor, B., Fiat, A., Naor, M., Pinkas, B. : Tracing traitors. IEEE Trans. Inform. Theory **46** (2000) 893–910
6. Schaathun, H. : The boneh-shaw fingerprinting scheme is better than we thought. IEEE Trans. Information Forensics and Security **1** (2006) 248–255

7. Peikert, C., Shelat, A., Smith, A. : Lower bounds for collusion-secure fingerprinting codes. In : Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Baltimore, MY, USA (2003) 472–479
8. Tardos, G. : Optimal probabilistic fingerprint codes. In : Proc. of the 35th annual ACM symposium on theory of computing, San Diego, CA, USA, ACM (2003) 116–125
9. Skoric, B., Vladimirova, T., Celik, M., Talstra, J. : Tardos fingerprinting is better than we thought. IEEE Tran. on IT **54** (2008) arXiv :cs/0607131v1.
10. Furon, T., Guyader, A., Cérou, F. : On the design and optimisation of tardos probabilistic fingerprinting codes. In : Proc. of the 10th Information Hiding Workshop. LNCS, Santa Barbara, Cal, USA (2008)
11. Cérou, F., Furon, T., Guyader, A. : Experimental assessment of the reliability for watermarking and fingerprinting schemes. EURASIP Jounal on Information Security **ID 414962** (2008) 12 pages
12. Xie, F., Furon, T., Fontaine, C. : On-off keying modulation and tardos fingerprinting. In : Proc. ACM Multimedia and Security, Oxford, UK, ACM (2008)