Fingerprint Orthogonal et Collusion (version publique)

Romain Vigoulette sous la direction de Stanislas Francfort
(France Télécom R&D MAPS/NSS)

Du 2 avril au 29 septembre 2006

Je remercie Stanislas Francfort pour m'avoir encadré et soutenu pendant le stage, Steven Prestwich pour son aide sur les BIBD et l'algorithme CLS, Andreas Westfeld pour ses conseils pour le challenge ECRYPT, tout le laboratoire NSS de Caen et en particulier Jacques Traore et Didier Guérin avec qui j'ai partagé mon bureau.

Table des matières

Première partie

Présentation de l'entreprise

S'il est une leçon essentielle à tirer de ce stage, c'est bel et bien de voir loin devant soi et de bien cerner les tenants et les aboutissants d'une mission. Or cerner les objectifs d'une mission ne peut se faire sans comprendre la stratégie du groupe. Ainsi nous brosserons à grands traits le portrait d'une entité, France Télécom R&D, qui, plus que jamais, doit faire figure de proue au sein du groupe France Télécom dans un contexte de concurrence acharnée entre les opérateurs téléphoniques.

0.1 Un peu d'histoire : le CNET

Fondé en 1944, le CNET (Centre National d'Études des Télécommunications) a contribué de façon déterminante aux progrès des services et des réseaux de télécommunications français, ainsi qu'à l'édification d'une industrie forte dans ce secteur. Initiateurs de paris technologiques audacieux, ses travaux ont abouti, en 1970, à l'installation du premier commutateur temporel en Europe. En 1978, eut lieu l'ouverture commerciale du premier réseau de données utilisant la commutation par parquets (Transpac), suivie, en 1981, par l'ouverture du service vidéotex Télétel célèbre par son terminal le Minitel. Ces réalisations ont connu un succès de réputation mondiale. Le CNET s'est aussi illustré parmi les pionniers des télécommunications spatiales. Il fut, en particulier, le concepteur et l'architecte des satellites Télécom 1 et Télécom 2. Il eut un rôle déterminant dans la définition et l'introduction du réseau numérique à intégration de services (Numéris). Grâce aux innovations du CNET et des GIE associés, la carte à mémoire a pu atteindre un niveau de maturité permettant la généralisation, en France, des Publiphones à carte. Ces innovations ont conduit au développement des systèmes ouverts de contrôle d'accès pour la télévision à péage et des nouveaux systèmes d'identification des utilisateurs de services numériques radiomobiles. Depuis le 1er mars 2000, le CNET poursuit ses activités sous le nom de France Télécom Recherche & Développement.

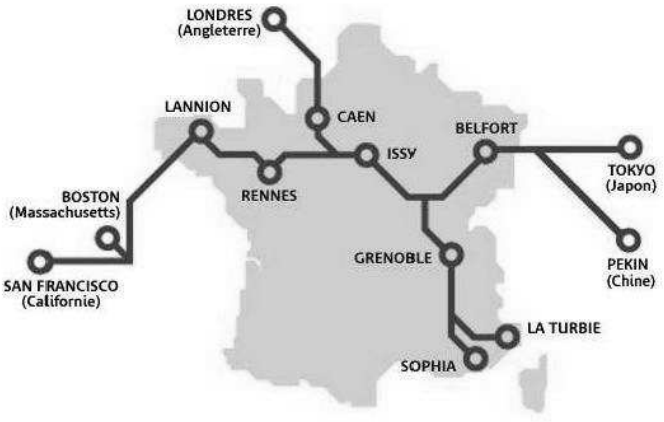
0.2 Missions

Les laboratoires de France Télécom Recherche & Développement sont à l'origine d'environ 70% des produits et services commercialisés par le Groupe. Avec près de 3700 chercheurs et ingénieurs, France Télécom Recherche & Développement est un atout majeur pour France Télécom. Ses missions consistent à anticiper les évolutions technologiques et les changements d'usages, innover pour offrir aux clients le meilleur des technologies, et explorer de nouvelles sources de croissance pour fournir au Groupe les assises d'un développement durable. Peu d'opérateurs dans le monde peuvent justifier d'un tel atout, pensé et organisé pour servir la stratégie globale de développement.
Avec ses 8 sites en France, plus deux aux États-Unis (San Francisco, Boston), un au Japon (Tokyo), un en Angleterre (Londres) et un en Chine (Pékin), il accompagne également le développement international de France Télécom en mettant son expertise au service de ses filiales étrangères, en s'impliquant auprès de grands groupes industriels, de la communauté scientifique mondiale et des instances internationales.
Pour répondre à ses ambitions, France Télécom Recherche & Développement

a accéléré sa réorganisation dès 1995 en se recentrant sur les domaines de recherche liés aux stratégies de marché de France Télécom, et en donnant la priorité à la recherche appliquée. En effet, lorsqu'on a détecté une opportunité d'innovation et déterminé sa pertinence, priorité est donnée à la recherche d'applications concrètes et à la réduction du cycle de développement. France Télécom Recherche & Développement poursuit son évolution aujourd'hui et améliore en permanence sa capacité d'innovation à la fois technologique, marketing, managériale et organisationnelle. Ainsi, le centre de recherche contribue également à développer le capital humain du Groupe, en jouant un rôle de vivier de compétences et en aidant les Ressources Humaines à anticiper les évolutions de métiers et les besoins en compétences.

0.3 Quelques chiffres

Voici un résumé des chiffres les plus significatifs, mettant en évidence l'importance du laboratoire de Recherche & Développement pour le groupe France Télécom :
· Plus de 3 700 collaborateurs, dont 3 000 ingénieurs, chercheurs et techniciens.
· 14 sites d'implantation en France, aux USA (San Francisco et Boston), au Japon
(Tokyo), en Angleterre (Londres) et en Chine (Pékin). (Voir Fig. 1)
· 7 094 brevets détenus au niveau mondial.
· 418 nouveaux brevets et 349 logiciels déposés sur les douze derniers mois. Un nombre de contrats de licences et de transferts de savoir-faire en augmentation permanente.
· 80% des clients satisfaits de la qualité des relations et de l'écoute.
· 200 recrutements chaque année.
· 150 thésards français et étrangers.
· 500 stages-terrain réalisés par les cadres de France Télécom Recherche & Développement dans les services opérationnels de France Télécom.
· Une quinzaine de start-ups créées depuis fin 1998 sur l'initiative d'ingénieurs de France Télécom Recherche & Développement, générant presque un millier



0.4 Structure

Une structure de la taille de France Télécom Recherche & Développement nécessite une organisation stricte est un découpage fin des différents pôles de recherche. Récemment la division Recherche & Développement a d'ailleurs subi une refonte complète de sa structure. Ainsi, au plus haut niveau les Recherches ne sont plus gérées par sept directions mais par six Centres de Recherche & Développement (CRD), regroupés autour de trois axes de recherches différents.
Deux CRDs sont voués aux services pour les clients :
· SIRP : Services Intégrés Résidentiels et Personnels
· BIZZ : Services aux Entreprises
Deux CRDs sont responsables d'activités d'intérêt commun :
· MAPS : Middleware et Plates-formes Avancées
· TECH : Technologies
Enfin, deux CRDs sont axés sur l'intégration des réseaux :
· CORE : Coeur de Réseau
· RESA : Réseaux d'Accès
Chaque CRD est ensuite découpé en laboratoires, eux même découpés en Unités de Recherche et Développement (URD). C'est à Caen, au sein du laboratoire de Sécurité des Services et des Réseaux (NSS), dépendant du CRD Middleware et Plates-formes Avancées que j'ai eu l'opportunité d'effectuer mon stage. Nous allons donc présenter brièvement le laboratoire, les différentes URDs qui y sont rattachées.

0.5 Une expertise technique de pointe

Dans le cadre de l'enrichissement de son offre de services, France Télécom doit ouvrir ses réseaux aux clients, aux prestataires et autres opérateurs, tant pour améliorer son offre que pour respecter les nouvelles législations. Le laboratoire de Sécurité des Services et des Réseaux a pour mission d'apporter son expertise technique en matière de sécurité ainsi que les briques de sécurité nécessaires afin d'assurer la pérennité des produits proposés par le groupe. Le laboratoire se divise en trois URDs à savoir :
· RMC, Sécurité pour les Réseaux, la Mobilité et la Confidentialité : cette URD étudie et améliore la sécurité du réseau sémaphore, du réseau intelligent, de tous les aspects de la mobilité, ainsi que des services liés à la confidentialité,
· SPR, Sécurité des Services, du Paiement et des systèmes Réparties : cette URD conçoit et met en place la sécurité des services offerts aux clients de France Télécom, la sécurité des paiements électroniques et évalue et met en oeuvre la sécurité des systèmes répartis,
· SII, Sécurité Intranet/Internet, URD qui évalue et améliore la sécurité des réseaux de type Internet ou Intranet pour les besoins de France Télécom et de ses clients.

Deuxième partie

stage

Chapitre 1

Introduction

S'assurer qu'un contenu numérique n'est pas redistribué illégalement après qu'il ait été délivré aux clients requiert souvent la capacité de tracer et identifier des entités impliquées dans la redistribution de contenu multimedia. Le fingerprinting est une technologie permettant de faire respecter les droits numériques à l'aide de labels uniques, connus sous le nom de fingerprint. Ces derniers sont insérés de façon uniques dans chaque contenus avant la distribution.
Ces fingerprints peuvent faciliter le traçage des coupables qui utilisent leur contenu pour des buts non voulus.

Pour protéger le contenu, il est nécessaire que les fingerprints soient difficiles à effacer. Pour les contenus multimedia les fingerprints peuvent être insérés en utilisant des techniques de tatouage robustes à diverses attaques.
Garantir l'utilisation appropriée d'un contenu multimedia n'est, cependant, pas un problème de sécurité traditionnel avec un seul adversaire. L'internet a permis aux adversaires d'être plus proche les uns des autres. Il est maintenant facile pour un groupe d'utilisateurs, ayant des versions marquées différemment d'un même contenu, de travailler ensemble et de monter collectivement des attaques contre les fingerprints. Ces attaques, connues sous le nom d'attaques par collusion multi utilisateurs, fournissent une méthode efficace pour atténuer chacun des fingerprints des utilisateurs et ainsi empêcher leur détection.
Une insertion ou un schéma d'identification mal conçu peut rendre vulnérable un contenu face à une collusion. Cette dernière peut en effet produire avec succès une nouvelle version du contenu sans aucune trace des fingerprints.
Ainsi, la collusion est une vraie menace à la protection de données. Nous voulons donc concevoir des fingerprints capable de résister aux attaques par collusion et ainsi d'identifier les coupables.

Dans la première partie, nous allons décrire les moyens à notre disposition qui permettent de générer des fingerprints résistant aux attaques par collusion. Ensuite, dans la seconde partie, nous expliquerons les limites de ces solutions, notamment lorsqu'il s'agit de distribuer un contenu à grande échelle (ex : DVD). C'est aussi dans cette partie qu'une solution sera proposée afin de pallier à ces problèmes.

Chapitre 2

Le fingerprinting

Le watermarking [1]

Le watermarking consiste à ajouter à un médium, en l'occurrence une image, une marque qui doit être suffisamment imperceptible pour ne pas détériorer le médium et suffisamment robuste pour être extraite même après une ou plusieurs attaques.

Qu'est ce qu'un fingerprint (et par extension le fingerprinting) ?

Le fingerprint est un tatouage qui est propre à chaque utilisateur d'une version d'un contenu. Si un contenu est distribué à plusieurs personne, alors chaque version de ce contenu sera tatoué avec un fingerprint correspondant à chaque personne.
Dans ce cas le tatouage est appelé fingerprint car c'est l'empreinte de l'utilisateur du contenu numérique dans lequel il a été inséré.
Le fingerprinting est le tatouage (l'insertion) d'un fingerprint dans un contenu digital.

Les attaques

Plusieurs attaques sont possible pour atténuer ou faire disparaître un tatouage. Il y a les attaques sur le signal : compression JPG, ajout de bruit,...etc les attaques géométriques : la rotation, le décalage,... etc et les attaques par collusion [1]. La plus puissante attaque et la plus facile à réaliser est la collusion par moyenne.
Le but de la collusion est d'atténuer, voire même de faire disparaître le fingerprint. Le principe est de combiner plusieurs images identiques qui ont toutes un fingerprint différent. L'image ne changera pas mais le fingerprint sera un mélange de tous les fingerprints participant à la collusion.

ex :

Bob veut utiliser son image de façon illégale. Son but : atténuer, voir même détruire son fingerprint. Il veut faire une collusion. Il récupère l'image d'Alice.

L'attaque se passe comme suit :

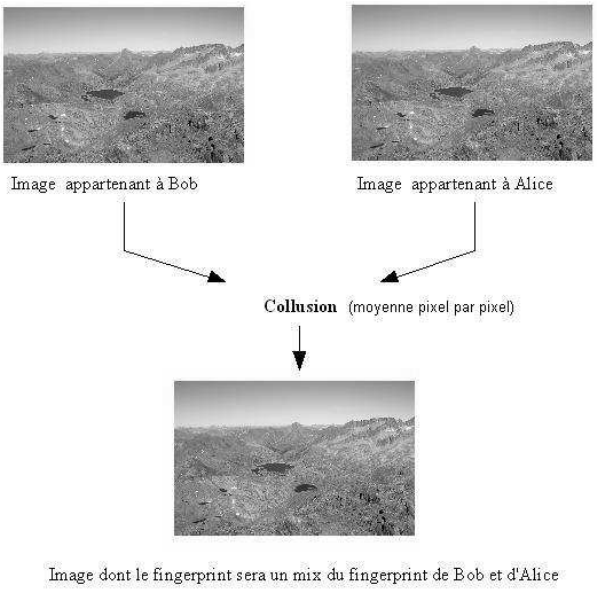


FIG. 2.1 – Principe de la collusion

Plusieurs méthodes existent pour réaliser une collusion : des méthodes linéaires et non linéaires. Dans le cadre de ce stage, nous allons étudier la méthode linéaire par moyennisation.

L'attaque par moyenne consiste à faire la moyenne, pixel par pixel, des n images participant la collusion.

$$U_{i,j} = \frac{1}{n} \sum_{k=1}^{n} A_{i,j}^{k}$$

(Où A^k est la $k^{ième}$ image).

Le principe de la décomposition en paquets d'ondelettes

Pour générer les fingerprints, nous utilisons une décomposition en paquets d'ondelettes.

La décomposition en paquets d'ondelettes à l'échelle 1 décompose l'image en quatre sous images. Chacune de ces images a pour taille la moitié de la taille de l'image originale. La première est un résumé de l'image originale : c'est à dire une version grossière (pixélisée). La deuxième met en valeur les détails verticaux, la troisième les détails horizontaux et la quatrième les détails obliques. A l'échelle 2, chacune de ces sous images devient l'image originale pour une nouvelle décomposition et ainsi de suite... On peut voir cette décomposition soit sous forme d'arbre, soit sous forme d'une image quadrillée par ses propres décompositions.
ex :

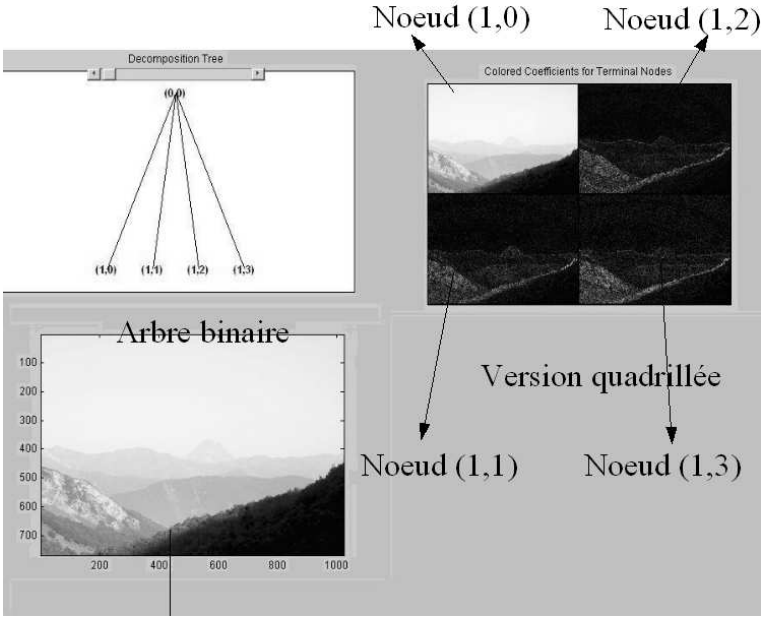


FIG. 2.2 – arbre de décomposition à l'échelle 1, version quadrillée de la décomposition et noeud (0,0) (ou image originale) en bas à gauche

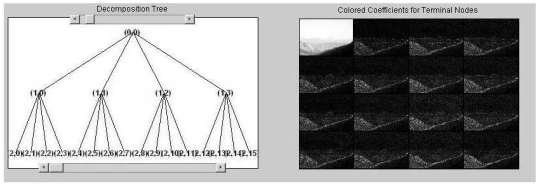


FIG. 2.3 – Même chose que la figure 1.2 mais à l'èchelle 2. Dans la version quadrillée il n'y a toujours que les noeuds terminaux c'est à dire les noeuds (2,0) à (2,15)

Un exemple d'application du fingerprint

Depuis quelques années, les films sont parfois disponibles en qualité DVD quelques semaines avant la sortie en salle. En 2002, lors de la cérémonie des Oscars, les DVD des films en compétition ont été délivrés aux 5803 membres éligibles de la MPAA (Motion Picture Association of America), votant aux nominations des Oscars. Le système de tatouage de Thomson a permis de leur distribuer des supports individuellement tatoués. Quelques semaines plus tard, les films suivants ont été trouvés sur internet : *The Last Samouraï, Something's Gotta Give, Mystic River*. Après avoir téléchargé les films comme n'importe quel utilisateur de logiciel peer 2 peer, l'acteur Carmine Caridi a été identifié comme la source de diffusion illégale. Il avait donné la copie à Russell Sprague qui l'a diffusé sur internet. Il a été exclu de la MPAA et a été condamné à 300.000$ de dommage et intérêts en novembre 2004. Le fingerprint montrait ainsi son intérêt dans la lutte contre la diffusion illégale de contenus numériques.

Chapitre 3

Fingerprinting orthogonal

3.1 La méthode de watermarking utilisée

Plusieurs méthodes existent pour insérer un watermark. Celle que nous allons utiliser est l'insertion additive. L'image à tatouer est consiérée comme une matrice, le tatouage aussi. Nous concevons le tatouage pour qu'il soit de la même taille que I. Ces deux matrices sont donc de même taille. Il suffit simplement de les additionner [4].

$$I + \delta w$$

Où I est l'image à tatouer, w le tatouage, et δ le coefficient d'atténuation.

3.2 Création d'un fingerprint

L'espace des fingerprints est un espace vectoriel engendrée par une base de bruit blanc orthogonaux entre eux : les U_i. Nous reviendrons sur la propriété d'orthogonalité plus tard.
Nous fabriquons une base de U_i. Nous concevons ensuite un code AND ACC (Anti Collusion Code) basé sur les BIBD (Balanced Incomplete Block Designs)). La taille de chaque mot du code est égale à celle de la base. Nous appliquons ensuite la formule (3.1) les U_i et un mot de code :

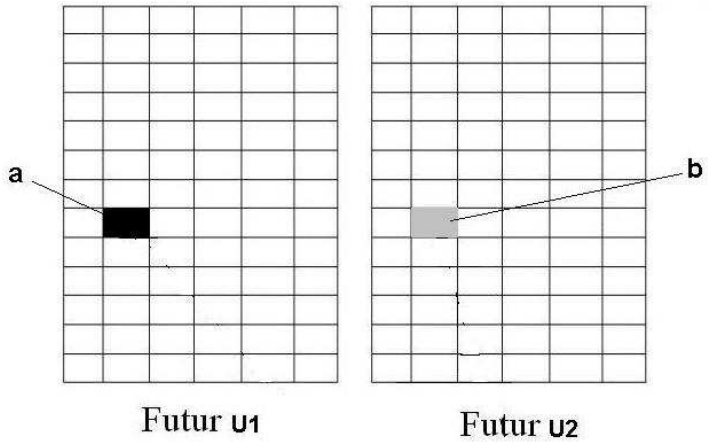
$$\sum_{i=1}^{n} (\pm 1) U_i \tag{3.1}$$

3.2.1 Construction d'une base orthogonale (U_i)

Nous choisissons de fabriquer les vecteurs de la base, U_i à partir d'une décomposition en paquets d'ondelettes d'une image nulle. Pour obtenir un U_i, nous allons donc partir d'une matrice nulle de la taille de notre image à tatouer I. nous avons calculé au préalable jusqu'à quelle échelle (voir chapitre 2) la matrice nulle sera décomposée (l'échelle est choisie de manière à obtenir des noeuds terminaux de $8 * 8$ pixels). Les noeuds sont tous à 0. Et le principe ici va être de mettre des 1 dans certains noeuds terminaux en faisant attention à ce que les U_i soient orthogonaux entre eux.

En résumé, les matrice nulles décomposées (c'est à dire les futurs U_i) vues en version quadrillée sont des matrices nulles. Pour qu'elles soient orthogonales entre elles il suffit de remplacer certains 0 par des 1 de manière a ce que le produit scalaire des matrices soit nul.

La figure suivante montre des décompositions quadrillées de deux matrices nulles (cela permet de voir tous les noeuds sous forme d'une matrice.



Pour avoir des U_i orthogonaux, il faut affecter des 1 dans des noeuds de U_1 qui ne pourront pas être affectés dans U_2 car le produit scalaire entre les deux matrices doit être nul(ex si nous décidons d'affecter a à 1, nous laissons b à 0).

Une fois que certains noeuds terminaux ont été remplis de 1, on fait la transformation inverse de la décomposition en paquets d'ondelettes.On a expliqué précédemment que les U_i décomposés à une certaine échelle sont orthogonaux entre eux (étant donné que l'on a fait en sorte qu'ils le soient) mais ce qui nous intérresse, c'est qu'ils soient orthogonaux entre eux une fois recomposés, dans leur forme définitive. Grâce au caractère orthogonal du splitting lemma, l'orthogonalité est conservée. On obtient donc bien des U_i permettant de créer des fingerprints orthogonaux.

3.2.2 Construction de codes anti collusions (AND ACC code) en utilisant les Balanced Incomplete Block Designs (BIBD)

Qu'est ce qu'un code anti collusion

Le fingerprint pour le $j$ème utilisateur, w_j, est construit en utilisant la formule (3.1) :

$$w_j = \sum_{i=1}^{n} b_{ij} U_i$$

où les coefficients $\{b_{ij}\}$, représentant le code du fingerprint, sont construits d'abord en concevant les mots de code mc de taille v avec des valeurs $\{0, 1\}$, et ensuite en utilisant la fonction suivante :

$$g(x) = \begin{cases} -1 & si \quad x = 0 \\ 1 & si \quad x = 1. \end{cases} \quad avec\ x = mc(i)\ \forall 1 \leq i \leq v \ (3.2)$$

Les codes anti collusions peuvent être utilisés avec la modulation de code pour construire une famille de fingerprints qui aura la capacité d'identifier les membres ayant participé à la collusion.
Un code anti collusion est une famille de mots de code pour lesquels les bits partagés entre les mots de code identifient de façon unique un groupe d'attaquants. Les codes ACC (Anti Collusion Codes) ont la propriété suivante :
le AND logique de tous les sous-ensembles de K, ou un peu moins, mots de code est unique. Cette propiété permet l'identification jusqu' à K attaquants. Un tel code est appelé : K-resilient AND Anti Collusion Code. *(resilient=résistant)*

Etudions maintenant un exemple simple de code ACC (un code 2-resilient). Les colonnes de la matrice C représentent les mots de code d'un ACC. Les mots de codes correspondent aux utilisateurs des fingerprints.

$$\begin{array}{lll} U_1 & \leftrightarrow & \\ U_2 & \leftrightarrow & \\ U_3 & \leftrightarrow & \\ U_4 & \leftrightarrow & \\ U_5 & \leftrightarrow & \\ U_6 & \leftrightarrow & \\ U_7 & \leftrightarrow & \end{array} \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} = C$$

$$\Updownarrow$$

$$\begin{array}{lll} w_1 & = & -U_1 - U_2 + U_3 - U_4 + U_5 + U_6 + U_7 \\ w_2 & = & -U_1 + U_2 - U_3 + U_4 + U_5 - U_6 + U_7 \\ & \vdots & \\ w_7 & = & +U_1 + U_2 + U_3 - U_4 - U_5 - U_6 + U_7 \end{array}$$

En examinant la matrice C, on voit que le AND logique de deux mots de code est distinct du AND logique de n'importe quels autres deux mots de code.
Par exemple, les w_1 et w_2 ci-dessus représentent les watermarks pour les deux premières colonnes de la matrice résultante de g(C). La moyenne $(w_1 + w_2)/2$

donne un vecteur qui a pour coefficient : $(-1, 0, 0, 0, 1, 0, 1)$. Le fait qu'un "1" apparait dans la cinquième et septième position identifie de façon unique les utilisateurs 1 et 2 comme membres de la collusion.

Comment les construit on ?

Un (v, k, λ)-BIBD est une paire (χ, A), où A est une collection de sous-ensembles (blocs) de k éléments tirés de l'ensemble χ de cardinal v.

(v, k, λ)-BIBD :
- v = taille de la base orthogonale
- b = nombre de bloques
- k = nombre d'élements par bloc.
- r : chaque élément de v a une occurence dans exactement r blocs.
- λ = nombre de fois qu'une paire d'élements se retrouve dans A.

Un (v, k, λ)-BIBD peut être utilisé jusqu'à $n = \lambda(v^2 - v)/(k^2 - k)$ utilisateurs, donc n blocs.

1	2	3	4	5	6	7
x	x	x				
x			x	x		
	x		x		x	
x					x	x
		x	x			x
	x			x		x
		x		x	x	

FIG. 3.1 – Exemple d'un code (7,3,1) BIBD. $= 1, 2, 3, 4, 5, 6, 7$ et $A = 123, 145, 246, 167, 347, 257, 356$. Ce schéma provient de [3].

Les indices des cases du tableau où il y a des croix correspondent aux indices de la matrice suivante où il y a des 0.

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Cette matrice représente en fait un code anti collusion 2-resilient où les colonnes sont les tatouages W_i de nos utilisateurs en fonction des éléments de la base orthogonale (lignes de la matrice). On dit qu'un code est K-resilient, avec $K = k - 1$, s'il résiste à la collusion de K attaquants. Ce type de code requiert $O(K\sqrt{(n)})$ vecteurs de base pour n utilisateurs.
On dit que ces codes issus de BIBD sont des AND-ACC (anti collusion codes).

Pour les tests, nous avons utilisé le code $BIBD-(16,4,1)$ suivant :

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1
\end{pmatrix}$$

La génération de BIBD est un problème NP-complet. Cependant il existe des solutions. Ici nous allons utiliser l'algorithme CLS (Constraints Local Search) basé sur une recherche locale [5] afin de générer des BIBD.
La recherche locale prend un point parmi un ensemble de points. Ces points doivent être définis par des contraintes locales. L'algorithme vérifie que chaque point satisfait à ces contraintes. Si ce n'est pas le cas pour l'un d'eux, alors ce dernier est déplacé et l'algorithme fait une revérification pour ce point.

Ici, nous décrivons une implémentation CLS pour la génération de BIBD. Les paramètres du BIBD sont (v, b, r, k, λ). L'algorithme va retourner une matrice de 1 et de 0 constituée de v lignes, b colonnes, r 1 par ligne, k 1 par colonnes, et un produit scalaire de λ entre chaque paires distinctes de lignes. Les paramètres d'un BIBD doivent satisfaire les conditions suivantes : $rv = bk, \lambda(v-1) = r(k-1)$ et $b \geq v$.
Pour construire nos paramètres :

- $\lambda = 1$ (toujours les code ACC (c'est ce qui permet l'unicité du traçage du logical AND de au plus K colonnes dans un code).)
- on choisit v tel que $v-1$ soit factorisable
- on factorise $v-1$ afin de trouver r et k car $(v-1) = r(k-1)$
- on déduit b de la formule : $rv = bk$

Dans l'algorithme, les contraintes à respecter sont les suivantes :

- $\rho_i^1 \leq r \ \rho_i^0 \leq b - r \ (1 \leq i \leq v)$
- $\chi_j^1 \leq k \ \chi_j^0 \leq v - r \ (1 \leq j \leq b)$
- $\pi_{ii'}^1 \leq \lambda \ \pi_{ii'}^0 \leq b - \lambda \ \left(1 \leq i \leq i' \leq v\right)$

Où :

$$\begin{aligned}
\rho_i^0 &= card\{j \in \{1 \dots b\} | m_{ij} = 0\} && (1 \leq i \leq v) \\
\rho_i^1 &= card\{j \in \{1 \dots b\} | m_{ij} = 1\} && (1 \leq i \leq v) \\
\chi_i^0 &= card\{i \in \{1 \dots v\} | m_{ij} = 0\} && (1 \leq j \leq b) \\
\chi_i^1 &= card\{i \in \{1 \dots v\} | m_{ij} = 1\} && (1 \leq j \leq b) \\
\pi_{ii'}^1 &= card\{j \in \{1 \dots b\} | m_{ij} = 1 \wedge m_{i'j} = 1\} && (1 \leq i < i' \leq v) \\
\pi_{ii'}^0 &= card\{j \in \{1 \dots b\} | m_{ij} = 0 \vee m_{i'j} = 0\} && (1 \leq i < i' \leq v)
\end{aligned}$$

Description :

- Initialisation d'une matrice de taille (v, b) à 2 (car $2 \neq 0 \neq 1$) exemple.
- On affecte chaque entrée de la matrice ligne par ligne, colonne après colonne.
- Pour chaque entrée de la matrice m_{ij} et un domaine de valeur $c = \{0, 1\}$, un compteur de conflits k_{ijb} est maintenu. Il compte le nombre de contraintes qui seraient violées si l'on affectait m_{ij} à c.
- Si k_{ijb} est égal à 0 alors nous affectons l'entrée m_{ij} sinon nous changeons l'affectation de m_{ij-1} et nous revérifions les contraintes pour m_{ij}. Si m_{ij} ne satisfait toujours pas les contraintes, on remonte à m_{ij-2} et on recommence.

En pratique il est difficile de créer de grands BIBD. Pour obtenir de grands BIBD, nous empilons de plus petits BIBD. Ce ne sont donc plus des BIBD mais nous allons le faire fonctionner quand même.
Exemple :

$$M = \begin{pmatrix}
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}$$

(3.3)

Sur la matrice (3.3) nous avons empilé deux (7,3,1) BIBD. La fonction (3.2) ne s'appliquera que sur les deux (7,3,1)-BIBD de la matrice : c'est-à-dire M_{ij} pour $1 \leq i \leq 7$ et$1 \leq j \leq 7$ et M_{ij} pour $7 \leq i \leq 14$ et $7 \leq j \leq 14$. Pour (3.3), il nous faut une base de 14 U_i. Les mots de code sont les colonnes de (3.3) et nous remarquons que les 7 premiers U_i feront partis des 7 premiers fingerprints (7 premiers mots de codes) et les 7 derniers U_i feront partis des 7 derniers fingerprints.

3.2.3 Génération des Fingerprints

Maintenant que tous les outils ont été définis, nous allons pouvoir générer les fingerprints. Nous prenons, par exemple le AND ACC code $7 * 7$ de (3.3) et nous remplaçons les 0 par des -1. Notre base de bruits blancs (vecteurs) orthogonaux doit être composée de 7 éléments (puisque le code est composé de 7 lignes) et les mots de code pour chaque fingerprint sont les colonnes. Nous n'avons plus qu'à appliquer la formule (3.1).

3.3 L'insertion

La technique d'insertion utilisée est *l'insertion additive*[1]. Après avoir préparé le fingerprint à insérer, nous allons insérer le tatouage dans l'image. Cette insertion est une modification des coefficients de luminance Y de l'image initiale. C'est donc avec la luminance que le fingerprint est ajouté avec un coefficient d'atténuation permettant de le rendre imperceptible à l'oeil humain. La formule utilisée est la suivante :

$$Y + \delta \sum_{i=1}^{n} (\pm 1) U_i \tag{3.4}$$

δ étant le coefficient d'atténuation (calculé pour que le psnr entre l'image tatouée et l'image non tatouée soit assez élevé). Dans l'insertion, trois carractéristiques sont importantes :

- La transformation RGB en YUV
- Le calul du psnr
- Le calcul du coefficient d'atténuation δ

3.3.1 La transformation RGB en YUV [2]

Un fingerprint peut être considéré comme une seule matrice. Une image couleur est répartie sur 3 matrices correspondant aux 3 couleurs *Rouge Vert Bleu* appelées RGB. Notre première idée a été d'ajouter le tatouage soit dans les *rouges*, les *verts* ou les *bleus*. Dans un souci d'imperceptibilité, l'idée suivante a été d'insérer le tatouage avec la luminance parce que les modifications y sont moins visibles par le modèle psycho-sensoriel.

Formule de la transformation RGB à YUV :

$$\begin{array}{rcrcl} Y & = & 0.299R + 0.587G + 0.114B & & \\ U & = & -0.169R - 0.331G + 0.500B & + & 128 \\ V & = & 0.500R - 0.419G - 0.081B & + & 128 \end{array}$$

Formule de la transformation YUV à RGB [2] :

$$\begin{array}{rcl} R & = & Y - 0.0009267 * (U - 128) + 1.4016868 * (V - 128) \\ G & = & Y - 0.3436954 * (U - 128) - 0.7141690 * (V - 128) \\ B & = & Y + 1.7721604 * (U - 128) + 0.0009902 * (V - 128) \end{array}$$

3.3.2 Le PSNR (Peak Signal to Noise Ratio)

Le psnr est une mesure de distance dans l'espace des signaux qui, dans ce cas va nous permettre de savoir si l'image tatouée est plus ou moins *éloignée* de l'image originale. Plus il est élevé, plus les deux images sont proches. Son unité est le décibel.
Notre but, lors d'une insertion, est d'avoir un psnr assez élevé pour que le tatouage soit imperceptible mais aussi assez faible pour que la détection du tatouage soit toujours possible.

Formule du RMSE (Root Mean Square Error) :

$$\sqrt{\sum(\sum(I_w - I)^2)/N} \tag{3.5}$$

I_w étant l'image tatouée et N, le nombre d'élements de I.

Formule du PSNR :

$$20 * \log_{10}(255/rmse) \tag{3.6}$$

3.3.3 Le coefficient d'atténuation

Le coefficient d'atténuation sert à contrôler la puissance d'insertion du tatouage dans une image. Si celui-ci est trop élevé, le tatouage est visible.
Formule du Coefficient d'atténuation :

$$\delta = \exp[\ln(255) - \ln(oldrmse) - \frac{p}{20}\ln(10)] \tag{3.7}$$

Où *oldrmse* est le rmse entre I et I_w non atténué et p est le psnr voulu.

Démonstration :

$$\begin{aligned} oldrmse &= \sqrt{\sum(\sum(I_w - I)^2)/N} \\ &= \sqrt{\sum(\sum(w)^2)/N} \end{aligned}$$

w est le fingerprint.
On veut obtenir un psnr de p db donc :

$$\begin{aligned} psnr : \quad 20 * \log_{10}(\tfrac{255}{\sqrt{\sum(\sum(\delta w)^2)/N}}) &= p \\ 20 * \log_{10}(\tfrac{255}{\sqrt{\sum(\sum(\delta^2 w^2))/N}}) &= p \\ 20 * \log_{10}(\tfrac{255}{\delta\sqrt{\sum(\sum(w^2))/N}}) &= p \\ 20 * \log_{10}(\tfrac{255}{\delta oldrmse}) &= p \\ \log_{10}(255) - \log_{10}(\delta oldrmse) &= \tfrac{p}{20} \\ \log_{10}(\delta) + \log_{10}(oldrmse) &= \log_{10}(255) - \tfrac{p}{20} \\ \ln(\delta) &= \ln(255) - \ln(oldrmse) - \tfrac{p}{20}\ln(10) \end{aligned}$$

$\Longrightarrow \delta = \exp[\ln(255) - \ln(oldrmse) - \frac{p}{20}\ln(10)]$

3.4 La Détection [4]

Dans cette partie, nous expliquons le problème de détection des attaquants quand le code AND-ACC est utilisé [4]. Nous expliquons notamment deux algorithmes de détection : le *Hard Thresholdind Detector* et l' *Adaptative Sorting Detector*. Ces détecteurs sont basés sur le vecteur de corrélation T_N, dont le $i_{ème}$ composant s'exprime par :

$$T_N(i) = y^T u_i / \sqrt{\sigma_y^2 . \|u_i\|^2} \tag{3.8}$$

Où y est l'image qui a subit l'attaque par collusion et σ_y^2 est la variance de y dont la formule est :

$$\sigma_y^2 = \frac{1}{N-1} \sum_{i=1}^{n} \sum_{j=1}^{m} (y_{ij} - \bar{y})^2 \tag{3.9}$$

Où n est le nombre de ligne de la matrice de y et m le nombre de colonnes.$n*m = N$ et :

$$\bar{y} = \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij}$$

Pour les deux types de détection décrites ci-après, la première chose à faire est de calculer T_N pour $i = 1, \ldots, v$ (v étant la taille de notre base orthogonale).

Nous allons étudier le Hard Thresholding Detector et l' Adaptative Sorting Detector puis les comparer.

3.4.1 Le Hard Thresholding Detector

La figure 3.2 décrit le Hard Thresholding Detector. Une fois les T_N calculés, nous obtenons Γ.
$\Gamma = (\Gamma_1, \Gamma_2, \ldots, \Gamma_v)$, où $\Gamma_i = 1$ si $T_N(i) > \tau$ ou 0 sinon. Etant donné le vecteur Γ, nous pouvons déterminer qui sont les coupables. En effet, les T_N nous permettent de savoir quelles sont les éléments de la base orhtogonale qui sont les plus/moins présents dans le fingerprint de y. Γ ne retient que les vecteurs qui ont la plus forte corrélation avec le fingerprint de y et le détecteur donne les utilisateurs qui utilisent tous ces vecteurs (ce sont les colonnes de la matrice qui ont des 1 aux lignes correspondant aux éléments de la base). L'algorithme retourne un vecteur Φ dont les 1 indiquent les coupables.

$\Phi \leftarrow 1$;
Définir J **comme l'ensemble des indices où** $\Gamma_i = 1$;
....**for** $t = 1$ **to** $|J|$ **do**
.....$j(t)$;
.....**Définir** e_j **comme étant la** $j_{ème}$ **ligne de C ;**
.....$\Phi \leftarrow \Phi.e_j$;
....**end**
return Φ ;

"." correspond à l'opération logique AND.

FIG. 3.2 – Algorithme : HardDetAlg(Γ,C)

3.4.2 L' Adaptative Sorting Detector

Sur la figure (3.3), $f(T_N|\Phi) = y^T W_c / \sqrt{\sigma_y^2 . \|u_i\|^2}$ avec W_c, la moyenne des fingerprints soupçonnés de participer à la collusion (les suspects sont donnés par Φ).

Description : L'algorithme commence par trier les T_N dans l'ordre décroissant afin de vérifier soupçonner en premier les fingerprints des U_i qui ont la plus forte corrélation avec le fingerprint de y. J contient les numéros i des U_i, donc on se servira de J par la suite pour définir les fingerprints suspects (i.e., ceux qui utilisent cet U_i (ligne 8 de l'algorithme)). On définit aussi ce qu'on appelle le *Likelihood* qui est en fait un coefficient de corrélation entre l'image tatouée y et la collusion des fingerprints suspects. Le AND logique de la ligne 9 permet de réduire les suspects à chaque tour afin de se rapprocher du/des propriétaires du fingerprint inséré. On calcule ensuite le *Likelihood* et le comparons au *Likelihood* du tour précédent. Si la différence des deux est supérieure à un certain *seuil* alors, c'est que le nouveau *Likelihood* est significativement inférieur au précédent et que donc les suspects précédents sont certainement les coupables, sinon on réduit le nombre de suspects et l'on recommence l'algorithme.

Pour l'implantation, un des problèmes rencontrés a été de trouver le *seuil* (de la condition if dans l'algorithme) car dans l'algorithme tel qu'il est décrit dans l'article [1], il n'en est nulle part mention. Ce *seuil* a été rajouté car parfois la corrélation $LL(i)$ est légèrement inférieure à $LL(i-1)$ alors qu'elle devrait être supérieure. Ceci est dû à la trop forte corrélation de certains U_i avec certaines images.

3.4.3 Tests

J'ai programmé tous les outils décrit ci-dessus afin de pouvoir les vérifier, savoir quel est le psnr qui nous convient le mieux, répondre à certaines questions du chapitre suivant et les modifier en vue de la finalité du projet.

Pour les tests utilisant l'Adaptative Sorting Detector, le *seuil* (de la condition if dans l'algorithme) est à $0,9$.

Trier les éléments de T_N en ordre décroissant et enregistrer les indices correspondants dans un vecteur J ;
Innitialisez Φ à 1 ;
$i \leftarrow 0$ **et calculer le** *Likelihood* $LL(i) = f(T_N \,|\Phi)$
Flag $\leftarrow$ **true**;
....**while Flag et** $i < v$ **do**
.....i = i+1;
.....j = J(i);
.....**Définir**e_j **comme la** $j_{ème}$ **ligne de C** ;
.....$\Phi_{up} \leftarrow \Phi.e_j$;
.....$LL(i) \leftarrow f(T_N \,|\Phi)$;
.....**if** $(LL(i-1) - LL(i)) <= seuil$ **then**
.....$\Phi \leftarrow \Phi_{up}$;
.....**else**
.....**Flag** $\leftarrow$ **False**;
.....**end**
....**end**
return Φ ;

FIG. 3.3 – Algorithme : AdSortAlg(T_N,C,y,W_i)

Les courbes de faux positifs et faux négatifs pour des tests réalisés avec des images $1024 * 768$ subissant des attaques par collusion de taille 2, 3, 4 , et 5 sont en annexes. Le code anti-collusion utilisé est un (16,4,1)-BIBD 3-resilient.

Pour le psnr, les tests ont montré que 20 db est trop faible. En effet, avec un tel psnr, Le tatouage est perceptible dans les applats (zone d'image où il y a très peu de variations de couleur).
Le psnr 40 db est trop élevé. Les corrélations entre le tatouage et les U_i sont trop faibles et donc les algorithmes de détection ne trouvent pas les couplables. Les valeurs 25 db et 30 db donnent de meilleurs résultats. Les détecteurs tracent tous les coupables et le tatouage est imperceptible. Tous mes tests ont été réalisés, par la suite avec ces 2 valeurs.

Pour la résistance à la collusion, les courbes de faux positifs et négatifs (en annexe) montrent que le code anti collusion 3-resilient résiste très bien aux attaques par collusion de taille au plus 3, que ce soit avec l'Adaptative Sorting Detector ou le Hard Thresholding Detector.

Troisième partie

Challenge ECRYPT BOWS

Chapitre 4

Introduction

ECRYPT (European Network of Excellence for Cryptology) est une initiative d'une durée de quatre ans lancée par des experts européens en cryptologie. Son but est d'améliorer la collaboration entre les spécialistes de la sécurité informatique, les cryptologues et les experts des mesures contre les contrefaçons. ECRYPT a commencé ses activités le 1er février 2004.

Au cours de ce stage, j'ai eu l'occasion de participer à un challenge organisé par ECRYPT intitulé BOWS (Break Our Watermarking System. L'adresse internet de ce challenge est : http ://lci.det.unifi.it/BOWS/. Le but étant de tester la résistance de l'algorithme de tatouage décrit dans l'article [12].
Trois images étaient disponibles sur le serveur de BOWS. L'objectif était de rendre l'information insérée illisible pour le détecteur tout en gardant une distance faible ($psnr > 30db$) entre l'image non attaquée et attaquée.

Pour tester la validité de notre attaque, nous avions à notre disposition un oracle (c'est à dire nous soumettions une image attaquée autant de fois que nous le voulions et la réponse était instantannée et nous disait si oui ou non le tatouage était encore présent et si le psnr était supérieur à 30 db). Quand au psnr, nous avions la possibilité de le calculer nous même étant donné que nous avions les images non attaquées et attaquées.
La référence de l'article donnant la description des algorithmes d'insertion et de détection était donnée.
Nous allons donc d'abord étudier l'article fourni [12] afin d'y repérer des faiblesses potentielles, puis nous décrirons les attaques effectuées.

imm_1.raw

imm_2.raw

imm_3.raw

FIG. 4.1 – Les trois images tatouées fournies.

4.0.4 Qu'est ce que le PSNR (Peak Signal to Noise Ratio)

Le psnr est une condition pour la réussite du challenge. Il doit rester supérieur à 30 db après l'attaque car si le psnr est trop faible, des dégradations dues aux attaques seront visibles sur les images . Le psnr est une mesure de distance dans l'espace des signaux qui, dans ce cas va nous permettre de savoir si l'image tatouée est plus ou moins *éloignée* de l'image originale. Plus il est élevé, plus les deux images sont proches. Son unité est le décibel.
Notre but, lors d'une insertion, est d'avoir un psnr assez élevé pour que le tatouage soit imperceptible mais aussi assez faible pour que la détection du tatouage soit toujours possible.

Formule du RMSE (Root Mean Square Error) :

$$\sqrt{\sum(\sum(I_w - I)^2)/N} \tag{4.1}$$

I_w étant l'image tatouée et N, le nombre d'élements de I.

Formule du PSNR :

$$20 * \log_{10}(255/rmse) \tag{4.2}$$

Chapitre 5

L'algorithme d'insertion : insertion informée [12]

Le tatouage d'un message dans une image utilise, ici, un algorithme d'insertion appelé : l'insertion informée qui se découpe en plusieurs étapes :

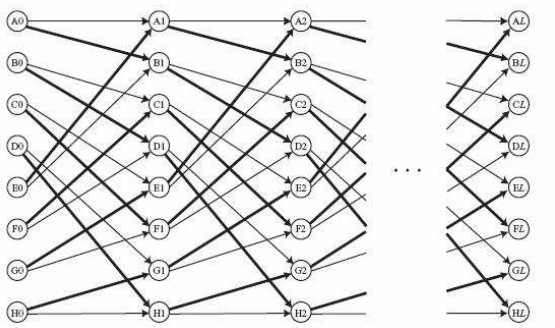
- Le codage du message appelé codage informé qui retournera un mot de code qui sera un vecteur de référence dans l'étape suivante.
- Le tatouage de l'image. Cette dernière est modifiée en fonction du vecteur de référence.

5.1 Le codage informé

Le codage informé permet de coder un message m en un vecteur de référence qui servira dans la phase d'insertion. On découpe l'image en bloc $8 * 8$, et la taille du message m à coder doit être égal au nombre de blocs.

Le treillis

Le treillis est un élément essentiel qui apparaît dans toutes les étapes : codage, insertion, détection.



Ce treillis est un graphe orienté composé d'arcs et d'états $(A_0, A_1, B_0...)$.
Les états A_0 à H_0 sont reliés, par des arcs gras et fins, aux états A_1 à H_1 qui sont eux même reliés aux états A_2 à H_2...
L'ensemble des arcs faisant le lien entre les états A_i à H_i et A_{i+1} à H_{i+1} est appelé un pas.
Le nombre de pas du treillis doit être égal à la taille du message m car un pas codera un bit de m.
Les arcs fins codent un 0 et les arcs gras un 1.
Chaque arc porte un label de 12 bits. Les labels sont les même à chaque pas : c'est à dire que par exemple le label de l'arc $A0 \rightarrow A1$ est égal à celui de l'arc $A1 \rightarrow A2$.

Le codage proprement dit

Le codage d'un message grâce au treillis est la recherche d'un chemin dans celui-ci. Un message à coder est une suite de bits et chaque pas du treillis code un seul bit. Si, par exemple, le message est 10 alors le treillis sera composé de deux pas dont le premier codera un 1 et le deuxième un 0. Pour se faire, on modifie le treillis en supprimant les arcs qui ne codent pas le bon bit dans chaque pas, comme dans l'exemple suivant :

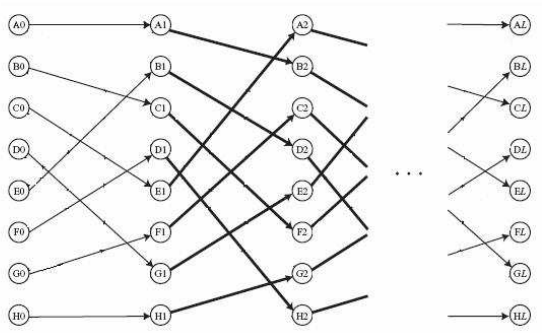


FIG. 5.1 – Pour un message $m = 011 \ldots ...0$ on obtient ce treillis.

En fait, le vecteur de référence sera l'un des chemins du treillis et donc sera l'ensemble des labels concaténés de ce chemin.
Pour savoir lequel sera le chemin retenu, on utilise l'algorithme de détection (voir chapitre suivant) entre le treillis et l'image à tatouer. Le vecteur de référence est choisi en fonction de l'image à tatouer. C'est pour cette raison que le codage informé s'appelle ainsi.

5.2 L'algorithme de détection

L'algorithme prend l'image en paramètre et va, dans un premier temps, la découper en bloc de $8 * 8$. Ensuite une DCT (Transformée en Cosinus Discrète) va être appliquée dans chaque bloc. Il va récupérer 12 coefficients DCT par bloc et les concaténer ensemble. Ces 12 coefficients sont toujours les même dans

chaques blocs. Ce sont les coefficients en gris sur la figure 5.2.

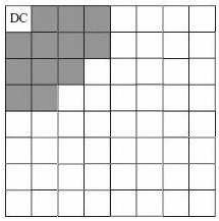


FIG. 5.2 – Les coefficients DCT modifiés sont ceux qui sont grisés en haut à gauche.

On obtient donc un vecteur v composé des coefficients DCT concaténés. La taille de ce vecteur est égal à la taille des chemins du treillis étant donné que chaque label de chaque arc est constitué de 12 bits et que le nombre de pas (donc d'arcs par chemin) est égale au nombre de bloc DCT(car le nombre de blocs est égal à la taille du message). On applique ensuite l'algorithme de Viterbi entre v et le treillis qui retourne le chemin qui a la plus forte corrélation avec v. C'est ce qu'on utilise pour trouver le vecteur de référence du chapitre (7.1).

5.3 L'insertion informée

On a donc vu que le vecteur de référence est choisi en fonction de l'image. Maintenant on va utiliser l'insertion informée : c'est à dire que l'on modifie l'image de façon à ce qu'elle se raproche le plus de ce vecteur. Les coefficients de l'image que l'on modifie sont les mêmes 12 coefficients DCT vu ci-dessus. Pour expliquer cette étape, on introduit le diagramme de Voronoï.
Ex :

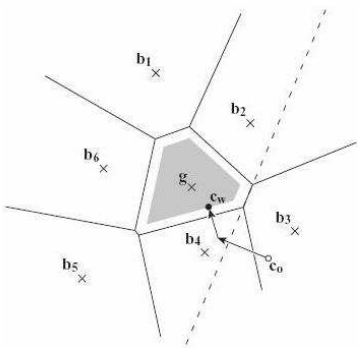


FIG. 5.3 – Diagramme de Voronoï où c_0 est l'image à tatouer, c_w l'image tatouée, les b_i sont les chemins, autre que le tatouage, possibles pour le message m et g est le vecteur de référence.

Le diagramme de Voronoï est composé de cellules dont la propriété est la suivante :

tout point d'une cellule a pour centre de cellule le plus proche, le centre de sa propre cellule.
Ceci nous permet donc de comprendre qu'il suffit de déplacer l'image à tatouer dans la cellule qui a pour centre le vecteur de référence (g sur la figure (5.3)).
L'image est alors considérée comme tatouée car si on utilise l'algorithme de détection entre cette image et le treillis, l'algorithme retournera le chemin de g. Cela signifie tout simplement que, d'après le diagramme, le centre de cellule (i.e. le chemin du treillis) avec lequel la corrélation sera la plus forte sera celui de la cellule où se trouve l'image.

Une fois que le vecteur contenant les 12 coefficients DCT de tous les blocs a été modifié, une permutation aléatoire de ce vecteur est effectuée. Cet aléa permet d'éviter de placer 4 coefficients modifiés consécutifs, car 4 coefficients consécutifs permettent de retrouver le chemin dans le treillis et de corriger d'éventuelles erreurs. Cet aléa sert à parer d'éventuelles attaques car il est difficile d'altérer 4 coefficients consécutifs sans dégrader l'image.
Lors de la détection, on applique la permutation inverse.

Chapitre 6

L'attaque

Le but de l'attaque est de rendre la détection inopérante, c'est à dire de faire en sorte que le détecteur ne renvoit pas le bon chemin et donc, pas le bon message.
Pour cela, on va concentrer l'attaque sur les 12 coefficients DCT (vu plus haut) car ce sont eux qui ont été modifiés. On ne va pas cibler tous les blocs. Le choix des blocs qui seront visés par une attaque se fait de manière aléatoire car on vise 4 coefficients consécutifs qui sont dispercés aléatoirement (à cause de l'aléa expliqué au chapitre précédent).
Après quelques tests, on se rend compte qu'aucune des attaques sur le traitement du signal (ajout de bruit, compression JPEG),et des attaques géométriques (rotation, décalage, ...) n'est efficace.
Pour monter une attaque efficace, il faut combiner certaines de ces attaques.
Nous avons plusieurs problèmes qui vont se poser car certaines de ces attaques, même combinées, n'auront aucun effet (JPEG, ajout de bruit). On ne peut donc pas choisir les attaques au hasard. Si une combinaison est susceptible de réussir, nous devons trouver comment paramétrer les attaques qui la compose : c'est à dire, par exemple, l'angle d'une rotation.
Etant donné que l'on a un oracle à notre disposition, on va utiliser la sensitivity attack de Linnartz [2] (expliquée ci-après), pour trouver le bon réglage, et d'abord voir si on peut monter une attaque efficace (avoir une combinaison qui aboutit à une attaque efficace).

Schéma de l'attaque

Pour effectuer les attaques on va utiliser le logiciel stirmark qui est un logiciel de traitement d'image (JPEG, filtre median, filtre gaussien, changement d'échelle, rotation,...) conçu pour tester la robustesse d'un tatouage. Il suffit de lui donner le réglage de la transformation voulue (angle pour la rotation par exemple).

Pour concentrer une attaque sur les 12 coefficients DCT de chaque bloc, le schéma mis en oeuvre (Fig 6.1) est le suivant :

1. duplication de l'image à attaquer : une que l'on va appeler A et l'autre B.
2. application d'une transformation avec Stirmark sur l'image B

3. découpage des images A et B en bloc de $8 * 8$
4. application de la DCT dans chaque bloc des images A et B.
5. reconstitution de chaque bloc composés des 12 coefficients DCT de B (paragraphe (7.2)) et des autres coefficients de A
6. application de la DCT inverse dans chaque bloc
7. concaténation des blocs qui donne l'image attaquée

2cm

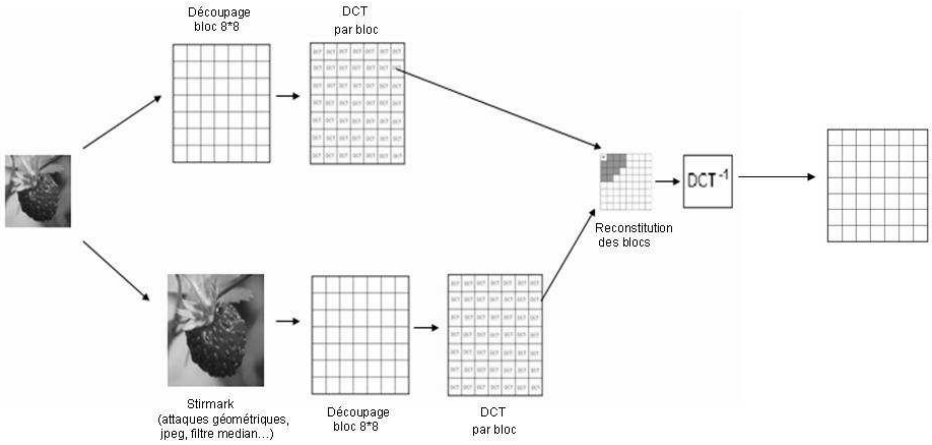


FIG. 6.1 – Schéma d'une attaque sur les 12 coefficients DCT de chaque bloc

Dans ce schéma, on a appliqué une seul attaque. Admettons que l'on veuille tester la combinaison suivante : une rotation et une décalage par la droite. On va appliquer la rotation avec le schéma décrit précédemment et on obtiendra une image C. Ensuite, on va appliquer le décalage, suivant le même schéma, à C.

Chapitre 7

Principe de la “sensitivity attack” de Linnartz [13]

Cette attaque n'est envisageable que si un oracle est à notre disposition.

Ici, pour attaquer une image, on modifie un vecteur v contenenant 12 coefficients DCT (ceux du chapitre 7.2) de certains blocs $8 * 8$ de l'image.
Soit n la dimension de l'espace de travail du watermark (cela peut-être les coefficients, le combre d'échantillon pour un son,...)
La sensitivity attack procède comme suit :

1. soit v' une transformation de v dans une direction aléatoire, avec une distance très éloignée
2. on teste l'oracle avec l'image attaquée contenant v. Résultat : ça ne marche pas. Si ça marche, on passe à l'étape 5.
3. on utilise un algorithme de dichotomie sur l'hyperplan de dimension $n-1$ passant par v et v', c'est à dire que la distance est divisée par 2.
4. on recommence à partir de l'étape 2 jusqu'à ce qu'une solution soit acceptée.
5. une solution est acceptée. Dans ce cas on a trouvé la frontière des solutions acceptées selon la dimension $n-1$. Pour améliorer l'attaque on continue avec l'hyperplan de dimension $n-2$ passant par la frontière qu'on a trouvé et perpendiculaire à l'hyperplan de dimension $n-1$.
6. on recommence à l'étape 2.
7. on peut continuer jusqu'à la dimension 0.

La sensitivity attack permet, dans notre cas, de trouver une bonne combinaison d'attaques et d'affiner les paramètres de réglage de ces attaques afin d'augmenter le psnr de l'image attaquée par rapport à l'image non attaquée.

Chapitre 8

Bilan et classement

Pour qu'une attaque soit validée comme ayant réussi, il faut que le détecteur n'ait pas réussi à lire le message m et que la distance (psnr) entre l'image originale et l'image ayant subie une attaque soit supérieure à 30 db

Après plus d'une semaine et demi de recherche intensive, nous avons réussi à trouver les combinaisons suivantes :
Pour imm-1.raw, on a utilisé :
- Une rotation d'angle -0.001°
- Un changement d'échelle (rotscale) de puissance 0.57

Résultat : psnr : 33.23 et le tatouage n'a pu être détecté.

Pour imm-2.raw on a ajouté un filtre median et la combinaison est :
- Une rotation à 0.04
- Un changement d'échelle à 0.05
- un filtre median à 2

Résultat : psnr :30.4 et le tatouage n'a pu être détecté.

Pour imm-3.raw :
- Un changement d'échelle à 0.05
- Un filtre median à 5

Résultat : psnr :30.15 et le tatouage n'a pu être détecté.

Le résultat est que la moyenne des psnr des 3 images est de 31.05 db.
Le classement est fait en fonction des moyennes des psnr des 3 images attaquées : du plus élevé au moins élevé. Notre équipe est $11^{ème}$ sur 29 ayant réussi le challenge.(http ://lci.det.unifi.it/BOWS/halloffame.php ?mode=VIEW)
En tout, plus de 400 participants ont testé leurs attaques.

Sans oracle, aucune attaque n'aurait aboutie car la bonne combinaison et le bon réglage des attaques est très dur à trouver lorsqu'on ne sait pas si le tatouage est encore détectable ou pas. Dans un contexte normal, c'est-à-dire un pirate voulant détruire la protection d'une image avec des outils classiques (ordinateur par exemple), il n'y a aucun moyen d'obtenir un oracle de ce type, a moins bien sûr d'avoir l'image hôte vierge de tout watermark.
L'algorithme de tatouage décri dans cette partie est donc très résistant.

Bibliographie

[1] Min Wu, Wade Trappe, Z. Jane Wang, and K. J. Ray Liu :*Collusion-Resistant Fingerprinting for Multimedia*

[2] http ://people.via.ecp.fr/ remi/ecp/tpi/rapport/yuv.html

[3] Stanislas Francfort :*Fingerprint et résistance à la collusion (http ://www.ufr-mi.u-bordeaux.fr/ yger/FrancfortFingerprint.pdf)*

[4] Wade Trappe, Min Wu, Z.Jane Wang, K. J. Ray Liu :*Anti-Collusion Fingerprinting for Multimedia*

[5] Steven Prestwich :*A Local Search Algorithm for Balanced Incomplete Block Designs*

[6] Jon Hamkins and Kenneth Zeger :*Asymptotically Optimal Spherical Codes*

[7] Jon Hamkins and Kenneth Zeger :*Asymptotically Dense Spherical Codes-Part I : Wrapped Spherical Codes*

[8] Jon Hamkins and Kenneth Zeger :*Asymptotically Dense Spherical Codes-Part II : Laminated Spherical Codes*

[9] ABBAS A.EL GAMAL, LANE A. HEMACHANDRA, ITZHAK SHERPLING, VICTOR K. WEI :*Using Simulated Annealing to Design Good Codes*

[10] Kari J. Nurmela :*Constructing Spherical Codes By Global Optimization Methods*

[11] http ://trond.hjorteland.com/thesis/node26.html

[12] Matt L. Miller, Gwenaël J. Doërr, Ingemar J.Cox :*Applying Informed Coding and Embedding to Design a Robust High-Capacity Watermark*

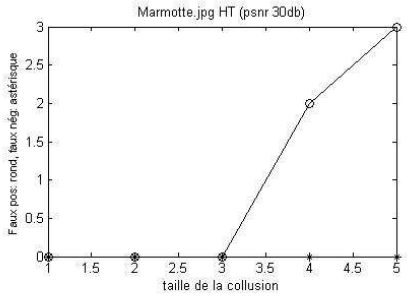
[13] Jean-Paul M.G. Linnartz and Marten van Dijk :*Analysis of the Sensitivity Attack against Electronics Watermarks in Images*

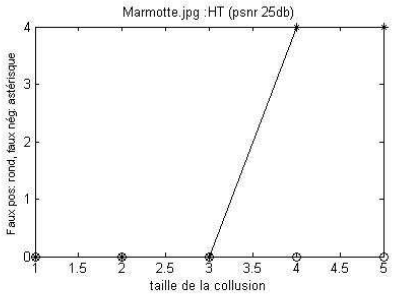
Quatrième partie

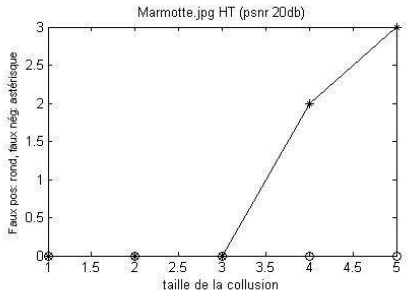
Annexe

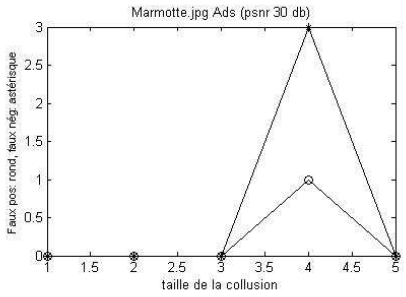
Courbes correpondant au chapitre 3.4.3.

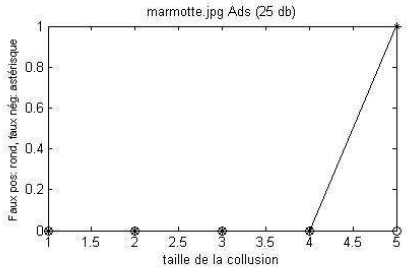
2cm

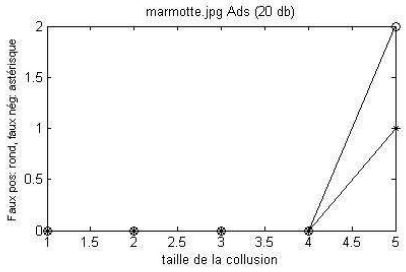


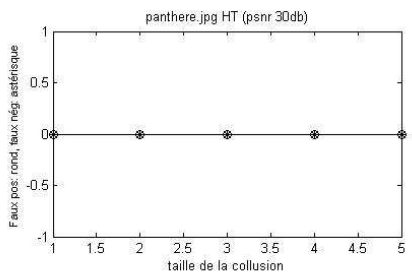


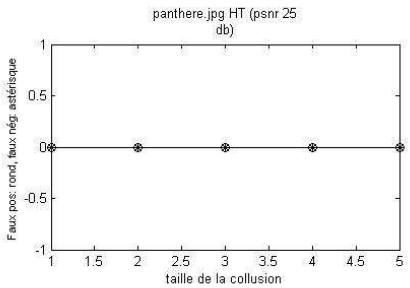


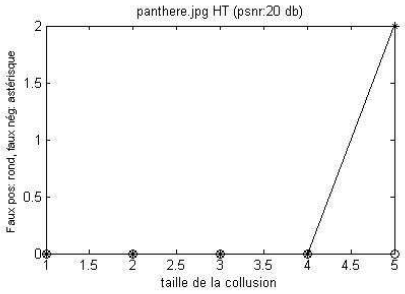


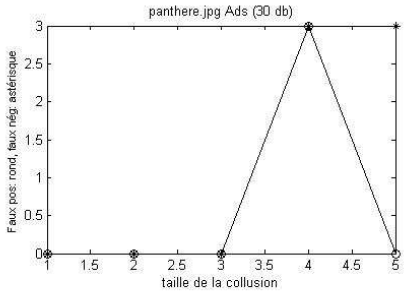


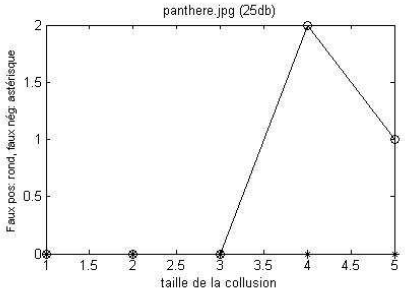


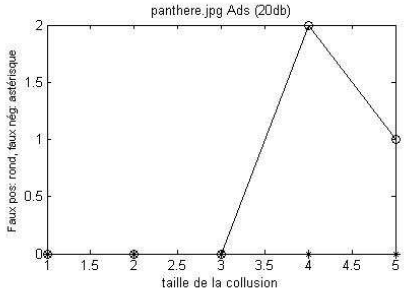


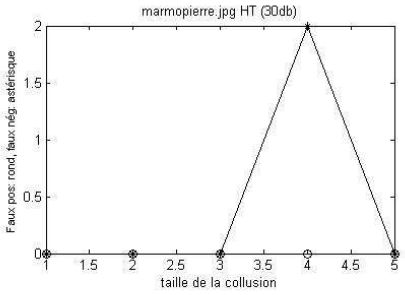


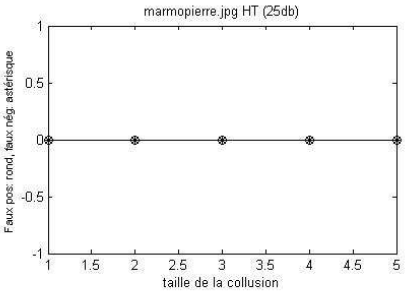

panthere.jpg HT (psnr:20 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

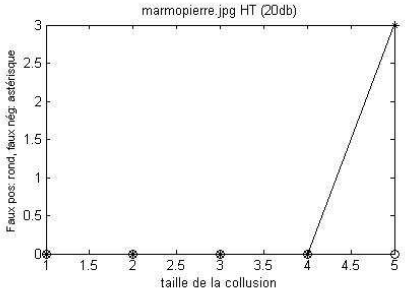

panthere.jpg Ads (30 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

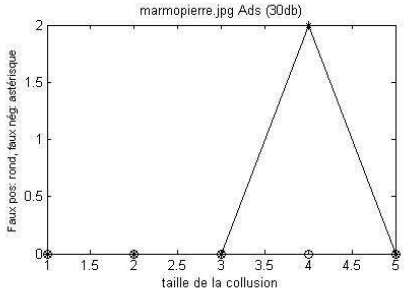

panthere.jpg (25db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

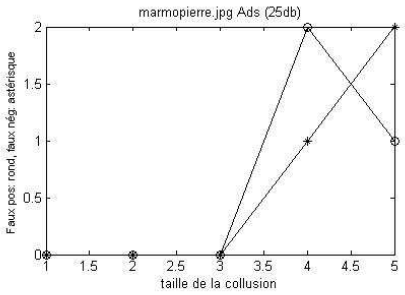

panthere.jpg Ads (20db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

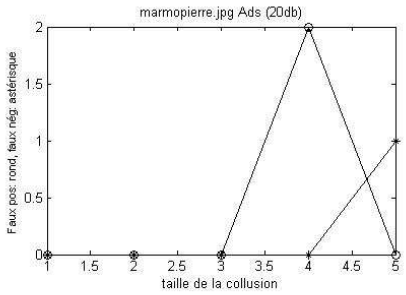

marmopierre.jpg HT (30db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

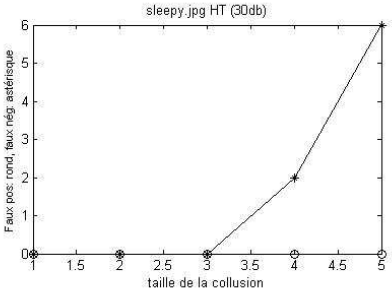

marmopierre.jpg HT (25db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

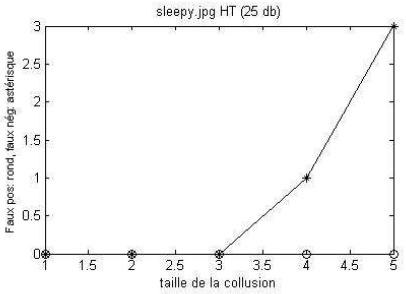

marmopierre.jpg HT (20db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

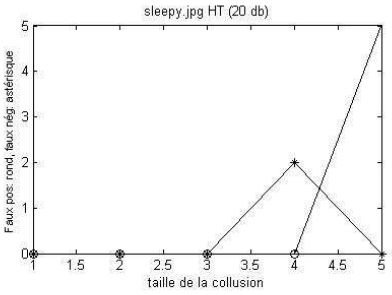

marmopierre.jpg Ads (30db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

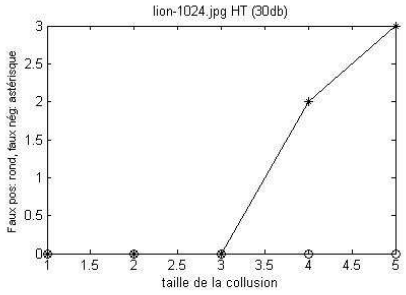

marmopierre.jpg Ads (25db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

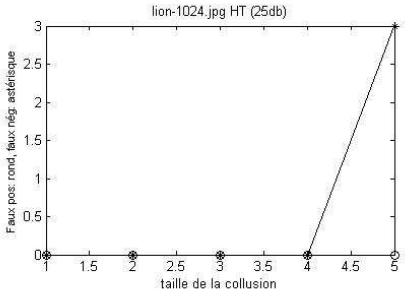

marmopierre.jpg Ads (20db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

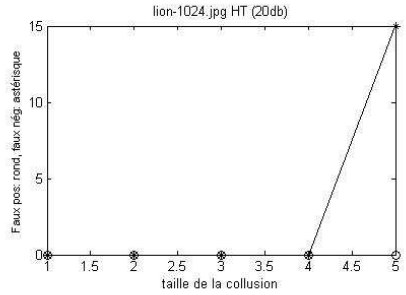

sleepy.jpg HT (30db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

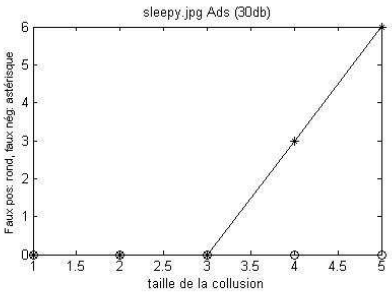

sleepy.jpg HT (25 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

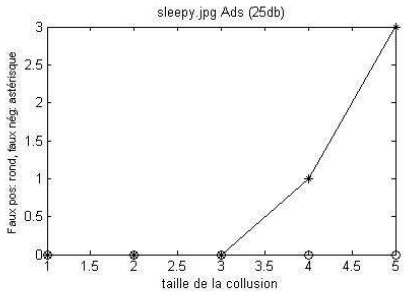

sleepy.jpg HT (20 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

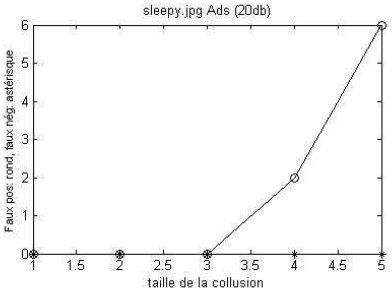

lion-1024.jpg HT (30db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

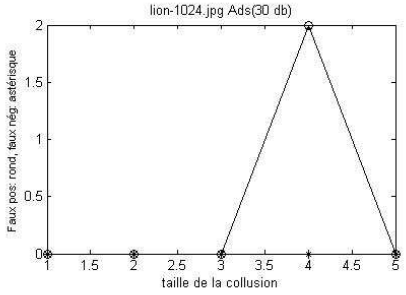

lion-1024.jpg HT (25db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

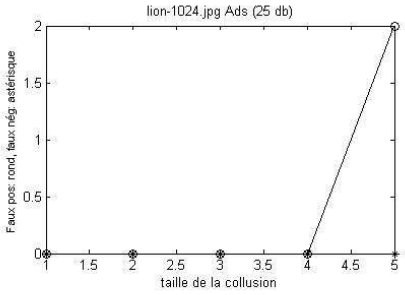

lion-1024.jpg HT (20db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

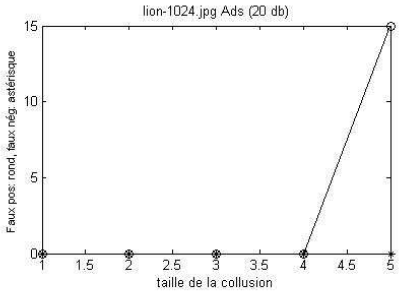

sleepy.jpg Ads (30db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

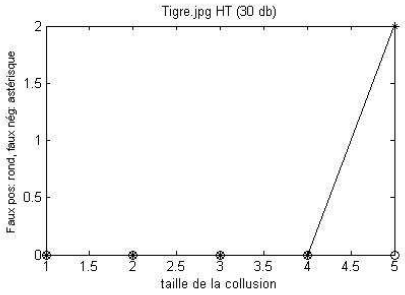

sleepy.jpg Ads (25db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

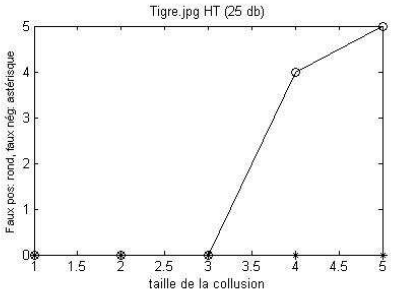

sleepy.jpg Ads (20db)
Faux pos: rond, faux nég: astérisque
taille de la collusion


lion-1024.jpg Ads(30 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion


lion-1024.jpg Ads (25 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion


lion-1024.jpg Ads (20 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion


Tigre.jpg HT (30 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion


Tigre.jpg HT (25 db)
Faux pos: rond, faux nég: astérisque
taille de la collusion

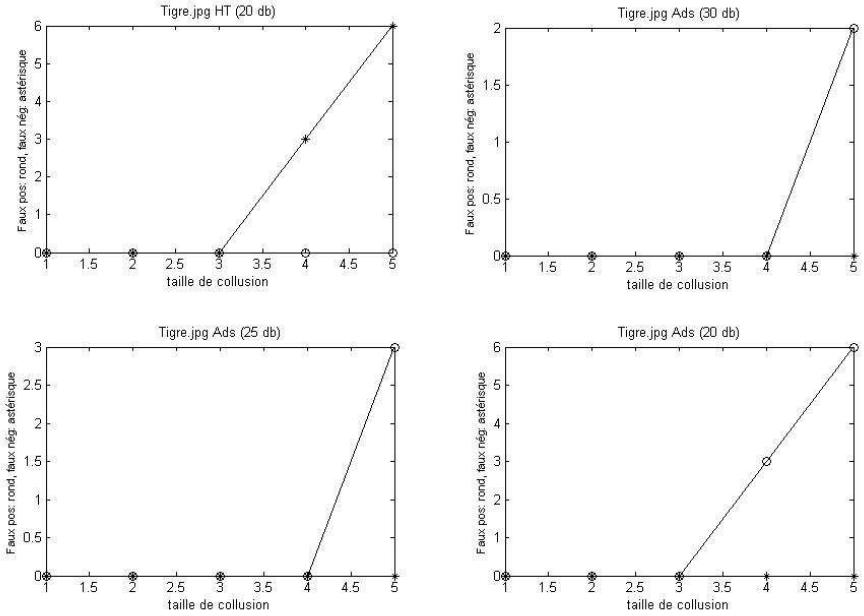


FIG. 8.1 – Résultats des détections sur plusieurs images. La taille des collusions est comprise entre 1 et 5 et le code ACC utilisé résiste à des collusions de taille au plus 3 (HT = Hard Thresholding Dectector Ads = Adaptative Sorting Detector faux pos = faux positif faux nég = faux négatif).