CONSERVATOIRE NATIONALE DES ARTS ET MÉTIERS

PARIS

MÉMOIRE

réalisé en vue d'obtenir

LE DIPLÔME D'INGENIEUR CNAM

en

Informatique

par

Meryem GUERROUANI

sous la direction de M. David GROSS-AMBLARD

Tatouage : application aux documents XML contraints

Soutenu le 28 juin 2005

JURY

PRÉSIDENT :	Professeur Bernard LEMAIRE Enseignant CNAM Paris Membre au Laboratoire CEDRIC
MEMBRES :	M. David Gross-Amblard Maître de conférences au CNAM Paris Encadrant, et membre au Laboratoire CEDRIC
	M. Mauvais Membre extérieur - Ingénieur
	M. Didier Ricois Membre extérieur - Ingénieur
	M. François-Yves Villemin Maître de conférences au CNAM Paris Membre au Laboratoire CEDRIC

Résumé

Le tatouage est une technique qui permet de marquer un document de façon indélébile. Les buts recherchés sont multiples, mais le plus important est la protection du droit de paternité de l'auteur. On dissimule dans le document des informations relatives à son propriétaire pour que personne ne puisse se l'approprier. Le tatouage est également utilisé pour identifier les utilisateurs malhonnêtes et éviter la piraterie. En effet un propriétaire d'un document qui veut commercialiser son oeuvre (par exemple un logiciel) peut marquer différemment tous les exemplaires autorisés (les différentes copies qu'il a vendues). S'il rencontre une copie illégale, le marquage lui permet de remonter à la personne responsable de la fraude, cette forme de tatouage s'appelle l'empreinte digitale ou le fingerprinting.

Plusieurs techniques de tatouage ont été élaborées dans le monde du multimédia comme la musique ou l'image, mais dans le monde des fichiers structurés comme les bases de données ou les documents XML, les algorithmes de marquage en sont à leurs prémices en raison de la complexité de leur structure et de leur dynamisme.

Dans ce mémoire nous présentons, dans un premier temps, l'état de l'art du tatouage des données structurées, ensuite nous étudions les travaux effectués par l'équipe Vertigo (laboratoire CEDRIC-CNAM) pour tatouer les bases de données relationnelles tout en préservant les résultats des requêtes d'utilisabilité prédéfinies. Nous généralisons ces travaux pour les étendre aux documents XML et introduisons la notion de la programmation linéaire pour résoudre les contraintes linéaires. Enfin nous faisons une expérimentation des méthodes employées sur des bases de données de grande taille.

Mots clés : base de données relationnelles, XML, tatouage, empreinte digitale, contraintes d'utilisabilité

Keywords : relational databases, XML, watermarking, fingerpinting, usability constraints

Travail soutenu par l'ACI Sécurité & Informatique - TADORNE 2004

Remerciements

Je tiens d'abord à remercier mon encadrant, monsieur David Gross-Amblard, de m'avoir fait confiance, pour ses précieux conseils, sa patience et surtout d'avoir cru en moi jusqu'au bout.

Je remercie monsieur le professeur Bernard Lemaire pour sa disponibilité et sa modestie et toute l'aide qu'il m'a apportée tout au long de mon parcours au CNAM.

Je remercie monsieur François-Yves Villemin pour son soutien, sa gentillesse et ses encouragements.

Je remercie tous les membres de l'équipe Vertigo qui m'ont accueillie avec le sourire et particulièrement monsieur le professeur Michel Scholl et madame Valérie Gouet.

Je remercie tous les membres de jury qui ont accepté d'être présents à ma soutenance.

J'ai une pensée affective à mon ami Jean-Paul Lebrun qui n'est plus de ce monde.

Pour terminer, un grand merci à ma famille adoptive en France la famille Amar, ma famille au Maroc, mon amie de toujours Ariane Bowando, ma soeur Nadia, Malika et Jean-Luc Hardouin pour tous les encouragements et l'espoir qu'ils ont placé en moi et qui ont su me faire rire dans mes moments difficiles.

Table des matières

Chapitre 1

Introduction

> "un tatouage est une amulette permanente, un bijou vivant qu'on ne peut enlever" (M.Tournier)

Le tatouage est une pratique ancestrale qui remonte à plusieurs civilisations. Le mot tatouage tire son origine du tahitien "TA-Tu", qui dérive lui-même de l'expression " TA-ATOUAS ", où "TA" signifie dessin et "ATOUAS" esprit. En effet, les indigènes marquaient leurs corps afin de se protéger contre les mauvais esprits et se concilier les grâces des bons esprits. Les romains, eux, utilisaient le tatouage pour marquer les esclaves et les criminels.

Aujourd'hui, l'avènement des technologies numériques et la démocratisation du Net font apparaître un souci majeur qui réside dans la facilité avec laquelle les documents numériques peuvent être copiés de façon illicite sans subir de détériorations. Ne pouvant pas faire la différence entre le document d'origine et les copies, il est très difficile, voir impossible, de protéger le droit de propriété intellectuelle. Pour pallier ce problème, des techniques de tatouage des oeuvres numériques peuvent être envisagées. Il s'agit alors de tatouer les oeuvres numériques comme faisaient nos ancêtres sur les corps humains.

1.1 Définition du tatouage

Une procédure de tatouage est formée d'une paire d'algorithmes. Le premier, nommé le marqueur, insère dans une oeuvre numérique un message secret. Cette insertion d'information doit respecter les contraintes suivantes :

– l'imperceptibilité : le tatouage est réalisé par altération des données. Cette altération doit être contrôlée pour que les données tatouées préservent leurs valeurs initiales.

– la robustesse : le tatouage doit subsister même si le document tatoué est transformé raisonnablement.

Le second algorithme, appelé le détecteur, est utilisé lors de la découverte d'un document suspect (une copie illicite par exemple). L'application du détecteur à ce document doit en révéler le message secret.

Dans la suite, on supposera posséder une telle procédure de tatouage, sans en préciser le fonctionnement, pour considérer ses applications.

1.2 Applications du tatouage

1.2.1 Protection des droits d'auteurs (copyright)

La protection des droits d'auteurs n'est pas un problème nouveau, et bien sûr, ne se limite pas au domaine informatique. Pour des raisons éthiques, légales et surtout morales, il est évident que le droit d'une oeuvre ou d'une création revient à celui qui la crée. En effet, pour produire une oeuvre, une personne fournit des efforts pour affiner sa création. Une fois cette oeuvre mise sur le Web, tout le monde peut la consulter. Malheureusement, des visiteurs malveillants et sans scrupules peuvent la télécharger, en faire des copies, et peuvent même prétendre qu'elle leur appartient. Le fameux problème lié à la copie et la distribution non autorisée de la musique qui fait la une des journaux ces derniers temps est le meilleur exemple pour illustrer le besoin du tatouage, puisque, en insérant sa signature dans son document, en cas de procès, le propriétaire peut prouver que le document en question lui appartient.

1.2.2 Authentification d'auteurs

Une personne physique s'identifie, en général, à l'aide d'une pièce d'identité afin qu'un tiers puisse vérifier que son porteur est

bien celui qu'il prétend être. Ou encore, quand une personne met une oeuvre en vente, elle y fait configurer ses coordonnées pour que des acheteurs potentiels puissent la contacter. Dans le monde numérique, le propriétaire insère dans son document une marque contenant des informations sur son identité. Si des clients sont intéressés par le document, alors ils peuvent en extraire la marque pour contacter le vendeur.

1.2.3 Intégrité des données

L'intégrité peut se définir par la question suivante : comment s'assurer qu'un document est authentique ou qu'il n'a pas été modifié après sa création ? Le tatouage peut garantir l'intégrité des documents, si la marque insérée contient une signature. Cette signature au sens cryptographique dépend du document et ne peut être contrefaite facilement. Si un document est modifié alors, la signature ne correspondra plus, et le document sera détecté comme non authentique.

1.2.4 Suivi des transactions (*monitoring*[1])

Le tatouage permet d'enregistrer ou contrôler tout trafic de données en insérant une marque dans le document original. Lors de la diffusion, la marque est automatiquement détectée. Ainsi, un détenteur de droits peut exiger des royalties en se basant sur la fréquence d'utilisation de son oeuvre. Il peut aussi, dans le cas où il a payé de la publicité à la radio par exemple, contrôler le nombre de diffusions.

1.2.5 Pister des transactions

L'idéal pour un vendeur est d'avoir un nombre maximum de clients achetant son produit, mais il lui est difficile, en cas de fraude, de retrouver le client à l'origine de la copie. Il est donc nécessaire de marquer les exemplaires autorisés avec des marques distinctes. Cette technique s'appelle *empreinte digitale* (*fingerprinting*). La marque contient alors l'identifiant du propriétaire et celui de l'acheteur. Autrement dit, si un fichier est vendu à n clients,

[1] *Monitoring* est certes anglophone mais figure dans le *Robert*.

alors, il y aura n copies contenant n marques différentes, relatives à son acheteur. Ainsi, si le propriétaire découvre une copie suspecte, il peut retrouver l'identité du client qui l'a achetée, en l'occurrence, la première personne responsable de la fuite.

1.2.6 Transmission de message secret

La stéganographie est la science de la communication d'information, par dissimulation dans une autre information. Cela permet de transmettre des messages cachés par un canal de communication sans être détecté. Ainsi, deux personnes qui veulent échanger des informations discrètement, cachent les messages dans un document quelconque sans que l'aspect extérieur de celui-ci ne soit modifié. Il s'agit d'une forme plus générale de tatouage, où le message secret marqué n'a pas forcément de rapport avec le document.

1.3 Problématique du tatouage

Comme introduit précédemment, le tatouage (*watermarking*) regroupe les techniques permettant d'insérer, de manière permanente et imperceptible, une information dans un support, sans dégradation apparente de celui-ci. Cette insertion ne se fait, évidemment, pas sans difficultés.

1.3.1 Altération du document

Concrètement, un document tatoué est un document dans lequel on a introduit intentionnellement des erreurs appelées marques. Ainsi, l'intrusion de la marque induit une modification volontaire du document d'origine. Deux grandes familles de modifications se présentent :

- les modifications syntaxiques : il s'agit ici d'altérer le format du document. Par exemple, dans le cas des documents textuels, il est possible de modifier imperceptiblement l'espacement entre les mots. Ainsi, un espace standard représentera par exemple la dissimulation d'un bit 0, deux espaces la dissimulation d'un bit 1. Dans le cas d'un document XML, on pourrait ajouter des balises supplémentaires dans le document comme par exemple <propriétaire> </propriétaire>. Les

méthodes syntaxiques ne sont pas très efficaces et rarement utilisées, car il suffit de supprimer les espaces en trop ou les balises pour effacer la marque. La suppression de ces données supplémentaires n'altèrerait pas le document copié.

– les modifications sémantiques : c'est une méthode plus robuste que la précédente, puisqu'elle agit sur le contenu même du document en y introduisant des distorsions volontaires. Pour un document textuel, la phrase " le musée du Louvre se trouve à Paris ", peut être transformée en " le musée du Louvre se situe à Paris ". Ou bien, pour des données numériques, si, par exemple, l'âge de Paul est de 10 ans, après la distorsion il sera de 11 ans.
Dans cette méthode, c'est bien la signification des données qui est altérée (la sémantique). La question qui vient alors immédiatement à l'esprit est : quelle est la limite de distorsion que l'on peut se permettre ?

1.3.2 Utilisabilité du document

Il est naturel de considérer que plus on peut altérer un document, plus le tatouage résistera aux attaques visant à l'effacer. Mais l'utilisation d'une marque de grande taille nécessite une forte altération du document d'origine. En d'autres termes, si la marque est très grande, le document risque d'être inutilisable. En revanche, si la marque est trop petite, il est plus facile de la détruire. Par conséquent, il faut trouver un bon compromis entre la taille de la marque d'une part, et le respect des critères d'utilisabilité du document d'autre part.

1.3.3 Robustesse et attaques

Un tatouage est dit robuste lorsqu'il résiste aux différentes attaques qu'on peut lui infliger. Il y a deux sortes d'attaques courantes contre lesquelles le tatouage doit résister :

1. Les attaques innocentes : une personne en possession d'un document tatoué peut être amenée à transformer le document pour son usage personnel. Pour une image, il peut la retailler, changer un peu ses couleurs, la compresser.

2. Les attaques volontaires (ou malveillantes) : un utilisateur malveillant soupçonne le document d'être tatoué. Il va alors tenter de supprimer la marque par des transformations volontaires. Il n'y a pas de limite à la forme de ces attaques. On peut citer par exemple :

   – Les attaques par altération aléatoire : l'attaquant, ne sachant pas où se trouve la marque précisémment, va modifier le document de façon aléatoire. Il espère ainsi altérer la marque et rendre le tatouage indétectable.

   – Attaque par suppression : un attaquant peut soit supprimer une partie du document tatoué car il a deviné l'emplacement de la marque, soit il supprime une partie du document qu'il a choisi aléatoirement en espérant que la marque s'y trouve. Si l'attaquant sous-estime la taille de la marque, alors son attaque échoue. En revanche s'il la surestime, alors la qualité du document sera dégradée. Forcément, puisque une grande partie de la donnée tatouée a été supprimée.

   – Attaque par surtatouage : un attaquant peut introduire sciemment des distorsions supplémentaires dans le document déjà tatoué en appliquant son propre algorithme de tatouage. Cela permet à l'attaquant de fabriquer une nouvelle marque imperceptible au-dessus de la marque initiale. Il peut ainsi se proclamer propriétaire du document. Pour répondre à cette attaque, un bon algorithme de marquage doit pouvoir retrouver la première marque apposée dans un document.

   – Attaque par collusion : dans le cas de l'empreinte digitale (*fingerprinting*), un groupe de clients possédant des copies autorisées (tatouées) va comparer ces copies. Comme chaque copie contient une marque différente, les clients en connivence peuvent découvrir l'emplacement de la marque. Ils possèdent ainsi une information précise sur la localisation de la marque, ce qui permet de l'altérer plus efficacement.

1.4 Cryptographie et Tatouage

La question qui revient fréquemment est la suivante : pourquoi ne pas utiliser des techniques cryptographiques ? Celles-ci ont fait leurs preuves pour la sécurité des documents et peuvent très bien

s'appliquer à la protection des droits d'auteurs. En effet, le propriétaire peut par exemple chiffrer le contenu de son document avant de le vendre. Seuls les clients ayant, légalement, acheté le document se voient attribuer une clé de déchiffrement.

Cependant, dés que le document est déchiffré, il n'est plus protégé. Nous voilà au point de départ. Autrement dit, un client malhonnête peut très bien acquérir, en toute légalité, un fichier protégé, le déchiffrer grâce à sa clé, et enfin, procéder à plusieurs copies qu'il peut distribuer librement et illégalement.

Par conséquent, il est nécessaire de trouver un complément ou une alternative à la cryptographie pour protéger les données, et cela même après leurs ventes. Le tatouage introduit une information (par exemple l'identité du propriétaire) qui suivra, de manière indissociable, la circulation de l'oeuvre.

1.5 Tatouage des bases de données et documents XML

Plusieurs techniques de tatouage ont été développées pour les documents multimédias comme les images, le son et la vidéo. Mais ces techniques ne peuvent pas être directement appliquées aux bases de données et aux documents XML, puisqu'il faut tenir compte des différences de caractéristiques entre les données structurées et les données multimédias.

En effet, un document multimédia, une image par exemple, est un objet statique sans structure particulière. Les données (par exemple la couleur des pixels) étant relativement indépendantes les unes des autres, il est facile de les altérer.

Par contre, une base de données relationnelle est définie par un *schéma* qui détermine sa structure. Chaque ligne d'une table d'une base de données (un n-uplet), est composée d'une suite d'*attributs*, dont le type du contenu est fixé. De plus, un sous-ensemble de ces attributs permet classiquement d'identifier de façon unique chaque n-uplet (on dit que la table possède une *clé primaire*). Enfin, le plus important dans une base de données est sans doute la réponse aux requêtes importantes pour l'application. Ces requêtes s'expriment au moyen d'un langage de requête comme SQL. Par exemple on souhaite :

– préserver les relations entre les différentes tables stockées dans une base, et s'assurer que le résultat d'une jointure reste inchangé avant et après le tatouage.

– Outre les relations entre les tables, il est nécessaire de préserver la cohérence des propriétés d'une même entité, c'est à dire, considérons, par exemple, une table décrivant des films qui se déroulent dans une salle de cinéma, contenant deux attributs temps : début et fin de séance. Il est impératif de s'assurer qu'après le tatouage, chaque tuple satisfait la condition suivante : le temps correspondant à la fin de la séance est plus tard que le temps correspondant au début de la séance.

Le même raisonnement peut être suivi pour un document au format XML. Pour ces documents, le type est fixé par une DTD (*document type definition)* ou plus récemment par une description *XML Schema.* Ces descriptions contraignent la forme du document considéré. De même, des requêtes importantes peuvent être spécifiées dans un langage comme *XPath* ou *XQuery*.

En résumé, pour des données fortement structurées comme les bases de données ou les documents XML, tatouer revient à trouver une altération raisonnable des données, qui préserve la structure du document et la plupart des réponses aux requêtes importantes.

1.6 Besoin d'une infrastructure

La prise en compte d'une base de données ou de documents XML de grande taille nécessite par essence l'utilisation de logiciels dédiés (système de gestion de bases de données, gestionnaire d'entrepôts XML natifs). Ces logiciels assurent l'abstraction des données, la vérification et l'évaluation optimisée des requêtes, le contrôle de la concurrence d'accès, la reprise sur panne.

Tatouer de tels documents nécessite également une assistance. On souhaite posséder une infrastructure logicielle permettant de :

– gérer des propriétaires et des acheteurs de données, identifiés par des clés ;

– stocker les documents originaux, et, pour chaque acheteur, fabriquer une version tatouée ;

– si un document suspect est découvert, utiliser un détecteur pour en déterminer l'éventuel propriétaire.

Comme les documents sont fortement contraints, l'infrastructure en question doit permettre de :

– spécifier quelles parties d'une base de données ou d'un document XML peuvent être modifiées par tatouage ;

– déclarer quelles sont les contraintes d'utilisabilité à préserver lors des opérations de tatouage. Ces déclarations doivent pouvoir se faire au moyen des langages de requêtes couramment utilisés, comme SQL ou XPath.

1.7 Organisation

La suite de ce mémoire est structurée comme suit. Le chapitre 2 présente l'état de l'art concernant le tatouage des documents structurés, comme les bases de données, les documents XML et les graphes valués. Le chapitre 3 donne le contexte de la réalisation du mémoire dans l'équipe de recherche Vertigo. Le chapitre 4 présente notre contribution propre, à savoir la généralisation de l'existant et son extension aux documents XML et aux contraintes linéaires. Le chapitre 5 donne nos expérimentations.

Chapitre 2

État de l'art

2.1 Le tatouage des bases de données relationnelles (R.Agrawal et J.Kiernan[2])

Agrawal et Kiernan fournissent une technique de tatouage adaptée aux bases de données relationnelles dont les caractéristiques sont :

- La marque n'est cachée que dans des attributs numériques sélectionnés par le propriétaire.
- L'utilisation d'une clé privée connue uniquement par le propriétaire.
- Le paramétrage de l'algorithme de tatouage par la clé privée pour déterminer l'emplacement de la marque, c'est à dire le tuple, l'attribut du tuple, l'indice du bit dans cet attribut, et enfin la valeur spécifique correspondant à ce bit.

Ce modèle se caractérise par les propriétés suivantes :

La détectabilité : face à une base de données suspecte, le propriétaire examine les m tuples qu'il a marqués, s'il trouve sa marque dans au moins n tuples (n<m fixé), alors cette base de données est suspecte.

La robustesse : le détecteur fonctionne même si une partie non négligeable des tuples a été modifiée.

Mise à jour incrémentale : dans le cas d'une modification (suppression, ajout d'un tuple, mise à jour) la marque doit être seulement recalculée pour le tuple.

L'imperceptibilité : la marque ne modifie pas de façon significative les valeurs des attributs.

Système aveugle : la détection de la marque n'exige ni la connaissance de la base de données originale ni la connaissance du filigrane.

L'algorithme de tatouage qu'ils proposent s'applique aux bases de données selon deux cas de figure :

- ***Cas d'une base de données avec clé primaire***

On suppose que v attributs sont candidats au marquage (seul le propriétaire décide des attributs qui peuvent être marqués et qui tolèrent le changement).

1. Appliquer une fonction de hachage à la clé primaire du tuple et la clé privée du propriétaire ($H(K_{prive}\, o\, H(K_{prive}\, o\, K_{primaire}))$)
2. Déterminer si le tuple sera marqué ou non en fonction de H et d'un paramètre β indiquant la proportion des tuples à tatouer (seul le propriétaire connaît β.)
3. Trouver les ϵ bits les moins importants (seul le propriétaire connaît ϵ).
4. Calculer l'indice du bit de l'attribut à marquer parmi les ϵ bits en fonction de H et de v.
5. Enfin, attribuer la valeur 0 ou 1 choisie selon le résultat de $H(K_{prive}\, o\, K_{primaire})$.

- ***Cas d'une base de données sans clé primaire***

S'il y a un seul attribut numérique, on partitionne les bits de cet attribut en deux parties. La première, constituée des bits de poids fort, sera considérée comme une clé primaire, et la seconde (les bits de poids faible) sera utilisée pour le marquage. Sinon, il faut choisir l'attribut le moins dupliqué pour éviter que la marque soit reproduite à l'identique plusieurs fois.

En ce qui concerne l'algorithme de détection, Agrawal et Kiernan proposent de définir un seuil n (nombre de tuples tatoués) au-dessus duquel, si le propriétaire retrouve n bits tatoués dans une base S, alors S est considérée comme piratée. Ils supposent également que la clé primaire n'a été ni supprimée ni modifiée car cela rendrait la base de données inutilisable.

Pour détecter la marque, l'algorithme doit, d'abord, déterminer le tuple, l'attribut et la position du bit qui a été marqué. Ensuite, il compare la valeur actuelle de ce bit à la valeur qui devait lui être attribuée au moment du tatouage. Une fois tous les attributs examinés, si le nombre de tuples contenant le filigrane est supérieur au seuil fixé, alors cette base de données est suspecte.

Cette technique n'altère que peu la valeur de chaque tuple. De plus, la moyenne globale des tuples est peu altérée. L'inconvénient de cette technique est qu'elle ne tient pas compte des contraintes d'intégrité de la base de données, ce qui peut nuire à son utilisabilité.

2.2 Le tatouage des bases de données relationnelles (R. Sion, M. Atallah et S. Prabhakar [3])

Dans cet article, les auteurs se basent sur une étude [7] effectuée sur le tatouage des ensembles numériques pour présenter une méthode de tatouage des bases de données, en contrôlant les propriétés qui doivent être préservées. Ils définissent une limite de distorsions autorisées des données en terme de mesure d'utilisabilité [8]. L'idée est de pouvoir modifier des valeurs d'attributs sans dépasser une certaine limite afin de préserver les propriétés initiales et la structure d'une base de données. Par exemple, pour un S = $\{s_1, s_2, ..., s_n\}$, ensemble de données à marquer, et V = $\{v_1, v_2, ..., v_n\}$, ensemble résultant du tatouage. On voudra vérifier que :

$$(s_i - v_i)^2 < t_i \qquad i = (1, 2, 3, ..., n).$$

L'insertion de la marque se fait en plusieurs étapes, à chaque fois qu'on introduit une distorsion à un attribut, on vérifie si toutes les conditions sont vérifiées, sinon on essaie sur un autre attribut.

Cette méthode est très générale car elle permet de tatouer en respectant un ensemble quelconque de contraintes. Cependant, le temps de calcul est très important, car pour chaque bit de marque inséré, il faut vérifier toutes les contraintes.

2.3 Le tatouage d'itinéraire (S. Khanna et F. Zane [4])

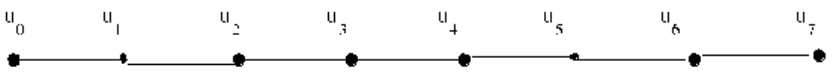
Une base de données d'informations cartographiques est représentée sous forme d'un graphe où les noeuds représentent des lieux, les arcs sont les liens entre les noeuds, et les poids sont les distances entre les noeuds adjacents.

Le but est de permettre à un propriétaire, d'une part, de vendre plusieurs copies de sa carte originale à différents fournisseurs d'accès qui répondent aux requêtes des utilisateurs finaux, et d'autre part, de pouvoir identifier chacun de ces fournisseurs. La technique consiste à introduire des modifications sur la longueur du chemin le plus court du graphe, mais il faut définir, auparavant, un seuil à ne pas dépasser, car une grande distorsion risque de dégrader la structure du graphe.

Les auteurs proposent trois types de modèles différents selon la nature de l'information A(u,v), fournie par le fournisseur d'accès en réponse à la requête (u,v) :

- **Modèle arête** : A(u,v) = $\ell(u, v)$ où ℓ est la longueur de l'arête (u,v).
- **Modèle distance** : A(u,v) = $d_G(u, v)$ où d_G est la distance entre les noeuds u et v.
- **Modèle chemin** : A(u,v) = *P*, où *P* est le plus court chemin entre les noeuds u et v.

L'exemple ci-dessous illustre la technique de tatouage proposée dans cet article pour un graphe G :



Soit X l'ensemble de noeuds u_i tels que i est impair, en l'occurence, X = $\{u_1, u_3, u_5, u_7\}$, et Y $\subseteq$ X l'ensemble des noeuds de degré 2, donc Y = $\{u_1, u_3, u_5\}$. Si $\mid Y \mid \geq \mid X \mid /3$, on peut créer des copies tatouées $Y' \subseteq Y$. Ainsi, une copie tatouée G' de G est obtenue en modifiant la longueur ℓ *en* ℓ' de la façon suivante :

$\forall\, u_{i_j} \in Y'$, $\ell'((u_{i_{j-1}}, u_{i_j})) = \ell((u_{i_{j-1}}, u_{i_j})) - 1$, et
$\ell'((u_{i_j}, u_{i_{j+1}})) = \ell((u_{i_j}, u_{i_{j+1}})) + 1$.

La figure ci-dessous montre le graphe tatoué G' :

Dans la suite de l'article, les auteurs se focalisent sur deux types de scénario : le scénario *sans adversaires* dans lequel un fournisseur d'accès, en possession d'une copie de la carte, répond correctement aux requêtes, et le scénario *avec adversaire* dans lequel le fournisseur peut introduire des distorsions supplémentaires aux réponses des requêtes afin d'éviter la détection.

Pour chaque scénario, une méthode de tatouage est obtenue avec une capacité de dissimulation garantie. Le tatouage est donc effectué en préservant une requête (plus court chemin). Des requêtes plus complexes ne sont pas étudiées dans l'article.

2.4 Travaux de l'équipe Vertigo[5]

Cet article présente la problématique du tatouage des bases de données relationnelles et des documents XML tout en préservant des contraintes spécifiques au langage. Ainsi un travail théorique a été effectué pour, d'une part, évaluer le nombre de tatouages différents que l'on peut insérer tout en préservant un ensemble de requêtes, et d'autre part, explorer la relation entre cette capacité de tatouage (nombre de tatouages distincts), la structure de la base de données (taille, relation entre les n-uplets), et la forme syntaxique de la requête.

Il est montré que, dans le cas général, il existe des requêtes à préserver très simples qui conduisent à une capacité nulle.

Exemple : Soit la requête à préserver suivante :

$\psi(x) = select\ sum\ (valeur)\ from\ G,\ V\ where\ G.id = x\ and\ G.id = V.idv.$

V :

idv	valeur
v_1	15
v_2	18
v_3	25
...	...
v_n	30

G :

id	idv
a	v_1
b	v_3
c	v_1
c	v_2
...	...

les valeurs impliquées dans cette requête sont représentées par la matrice suivante :

marque	+1	-1	+1	...	-1	
valeurs	v_1	v_2	v_3	...	v_n	distorsion
$\psi(a)$	1	0	0	0	0	+1
$\psi(b)$	0	0	1	0	0	+1
$\psi(c)$	1	1	0	0	0	0
...	...	...	...	...	...	...

La représentation graphique de cette matrice serait :

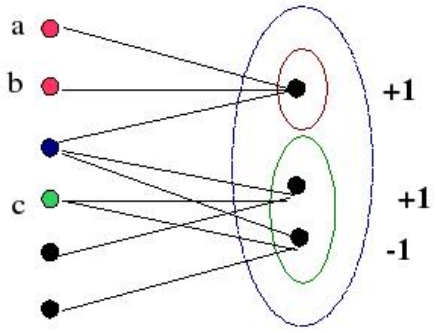


Figure 2.1

D'après ce graphe, on peut aisément constater que plusieurs valeurs impliquées dans la requête ψ subiront une altération de +1, ce qui implique une grande distorsion sur la somme (la requête ψ) de ces mêmes valeurs après le tatouage.

En revanche, si la base de données est de "degré borné" (par exemple si la base représente un graphe, son degré doit être borné), il y a toujours une bonne capacité de tatouage, quelle que soit la requête à préserver (SQL, XPath).

Exemple :

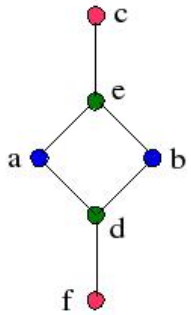


Figure 2.2

On peut montrer que la plupart des requêtes qui nous intéressent (SQL) ne font pas la différence entre les n-uplets de la base ayant des relations isomorphes avec les autres n-uplets, trois types de voisinages distincts sont présents dans l'exemple ci-dessus. En effet, en notant N(s) le voisinage d'un sommet s, on a N(a)≈N(b), N(e)≈N(d) et N(c)≈N(f). Deux requête ayant des paramètres isomorphes vont alors utiliser les mêmes attributs modifiables. Par exemple, si une requête ayant pour paramètre « a » utilise les valeurs d et e, alors une requête ayant pour paramètre b, puisque N(a)≈N(b), elle utilisera exactement les valeurs d et e, donc a et b sont dans la même classe d'équivalence. Ainsi à chaque équivalence, on associe une classe, et les classes de la figure 2.2 sont a, c et d. La matrice correspondante est alors comme suit :

marque	+1	-1					
sommets	a	b	c	d	e	f	distorsion
classes							
a				1	1		0
c					1		0
d	1	1				1	0

Le problème principal est alors de trouver plusieurs paires distinctes d'éléments pour leur appliquer les modifications (+1,-1) afin d'obtenir une distorsion globale égale à 0.

En conclusion, l'algorithme de tatouage présenté dans cet article est efficace, mais ne constitue pas une solution complète, car il n'est pas aveugle (il faut posséder la base d'origine pour faire la détection).

Chapitre 3

Contexte du mémoire

3.1 Introduction

Cette section présente le contexte général du mémoire, ainsi que le travail déjà effectué par Camélia Constantin[1] pour élaborer une technique de tatouage des bases de données relationnelles tout en préservant des contraintes SQL définies au préalable.

3.2 Architecture 3-tiers

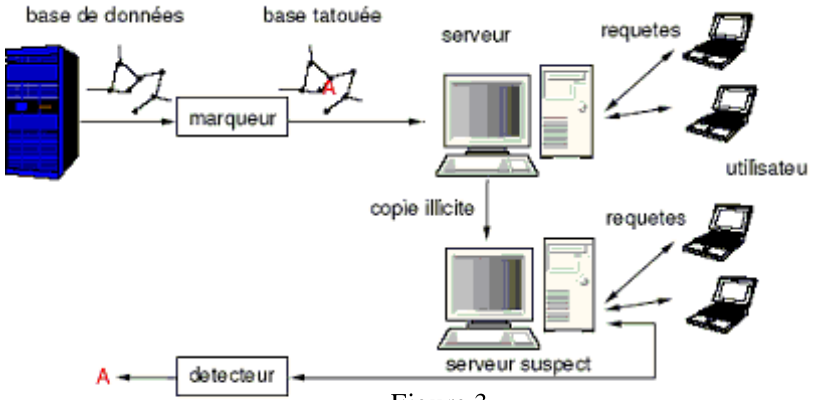


Figure 3

La figure 3 illustre un scénario à trois acteurs et deux algorithmes.

[1]Mémoire Ingénieur de l'université polytechnique à Bucarest réalisé au laboratoire CEDRIC, CNAM, Paris, dans l'équipe Vertigo.

Le propriétaire vend une copie de sa base de données à un serveur. Afin d'obtenir une copie tatouée, il applique un algorithme d'insertion de la marque appelée *marqueur*, qui consiste à apporter un ensemble unique de modifications à cette copie.

Le serveur achète cette base de données, et répond aux requêtes de plusieurs utilisateurs finaux à travers une interface (ex. page Web).

L'utilisateur final utilise l'application fournie par le serveur dans le but de faire un certain nombre de requêtes à ce serveur.

Le marqueur est l'algorithme d'insertion de la marque. Comme cette insertion modifie des valeurs d'attributs dans la base de données, le résultat des requêtes réalisées dans cette base peut être altéré. Par conséquent, l'insertion de la marque doit contrôler la modification de certaines requêtes pour que le résultat de ces requêtes reste acceptable par rapport à une limite donnée. Quand un serveur veut acheter une instance de la base de données, il passe un accord avec le propriétaire de cette base sur un ensemble de requêtes prédéfinies sur lesquelles il impose certaines distorsions. Le propriétaire doit garantir que la version tatouée de la base de données qu'il va vendre au serveur respecte bien les distorsions imposées. Bien sûr, le propriétaire peut rejeter la demande du serveur.

Le détecteur est l'algorithme de détection de la marque. En fait, le propriétaire agit comme un utilisateur final car il n'a pas un accès direct à la base de données suspecte, c'est à dire, il envoie des requêtes au serveur, et analyse les réponses pour détecter la marque.

Exemple 1 : on considère un exemple d'application concret pour illustrer les notions citées ci-dessus. Soit une compagnie possédant toute une série d'informations sur les pays européens[9]. Ces informations sont stockées dans une base de données appelée Europe. Une instance de cette base de données est représentée par des relations décrivant les fleuves et la démographie des pays européens :

La table pays :

idp	nomp	capitale	superficie(km^2)	population	densite
49	Allemagne	Berlin	356910	82 414 000	230.9
33	France	Paris	547026	59 850 000	109.4
34	Espagne	Madrid	504782	40 977 000	81.1
353	Irlande	Dublin	70 280	3 911 000	55.7
41	Suisse	Berne	41 285	7 000 000	177.0

La table fleuve :

idf	nomf	idsource	longueur(km)
83	Rhone	41	880
84	Garonne	34	650
85	Rhin	41	1320
87	Danube	49	2850
89	Loire	33	1022

Une entreprise qui veut créer une application pour établir des réseaux fluviaux ou organiser des croisières, peut être intéressée par ces données. Elle décide d'acheter cette base de données, ainsi qu'un ensemble de requêtes permettant de déterminer l'itinéraire et d'évaluer les coûts de déplacement afin de l'intégrer à son logiciel. Enfin, cette entreprise met cette application à la disposition des clients (utilisateurs finaux) qui désirent réaliser des requêtes pour obtenir des informations sur l'Europe ou acheter des billets.

3.3 Caractéristiques

Une distorsion raisonnable

Les valeurs d'attributs ne peuvent pas être modifiées arbitrairement, sinon elles deviennent insensées. En prenant l'exemple 1, si les distorsions sont introduites, d'une façon fantaisiste sur les valeurs de l'attribut $longueur$ de la table $fleuve$ et si la longueur du Danube, par exemple, devient égale à 700km après le tatouage, il est flagrant que c'est une information erronée puisque tout le monde sait que le Danube est le deuxième fleuve européen.

L'invisibilité de la marque

Il est évident que si la valeur dans laquelle la marque a été insérée est identifiée (ou localisée), ou si la marque est visible, il très facile pour un attaquant de détruire cette marque.

Malheureusement la définition des limites de distorsion ne tient pas compte des propriétés liées aux données. Ces propriétés représentent des contraintes que l'algorithme de l'insertion de la marque doit absolument respecter. Les contraintes qui ne peuvent pas être ignorées sont :

- **l'unicité** : chaque valeur modifiée par le tatouage doit être identifiable par une valeur unique qui est la clé primaire. Cette clé primaire doit être unique et inchangeable.
- **relation entre les attributs :** certaines relations entre les attributs doivent rester les mêmes avant et après le tatouage.
  Exemple :
  Dans la table $pays$, les attributs $superficie$, $population$ et $densite$ sont étroitement liés par la relation :
  $densite \approx population \div superficie$. Cette relation doit être respectée après le tatouage.
- **même résultat :** une requête doit avoir le même résultat avant et après le tatouage.
  Exemple :
  La requête « *select idf, nom from fleuve where longueur < 1000* »
  doit répondre (83, Rhone) et (84, Garonne) avant et après le tatouage.
- **préservation de jointure :** il est impératif de s'assurer que les tuples joints avant le tatouage, le sont également après le tatouage.
  Exemple :
  Après le tatouage de la base de données $Europe$, il faut s'assurer que les valeurs de l'attribut $idsource$ de la table $fleuve$ correspondent aux même valeurs des attributs idp de la table $pays$ avant le tatouage.

Détectabilité

Lorsqu'une base de données est suspecte, le propriétaire doit pouvoir identifier le client à qui la base de données a été vendue. Ainsi, la technique de tatouage utilisée est l'empreinte digitale (fingerprinting) : l'algorithme insère une marque différente pour chaque client qui achète la base de données.

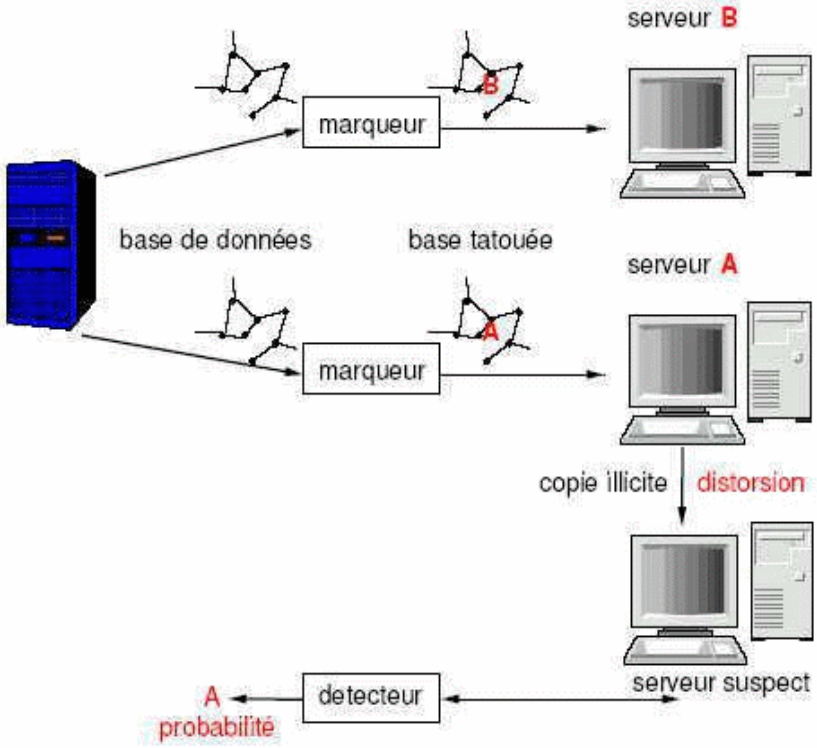


Figure 3.2

La figure 3.2 montre un exemple de tatouage d'empreinte digitale.

Système basé sur des clés

Selon Kerchoffs[2], la méthode utilisée pour l'insertion de la marque doit être publique. Par conséquent, la robustesse de la marque est liée au choix de la clé privée qui est connue uniquement du propriétaire.

Système aveugle

la détection de la marque ne requiert pas l'accès à la base de données originale. Si un propriétaire suspecte un serveur d'utiliser une copie illicite de sa base de données, alors il se comporte comme un

[2]Deux axiomes fondamentaux d'un algorithme cryptographique : "il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi" et "la clé doit pouvoir en être communiquée et retenue sans le secours de note écrite, et être changée ou modifiée au gré des correspondants".

utilisateur final. En analysant les résultats des requêtes préservées, il peut savoir si la donnée contient ou non sa marque. C'est pourquoi, l'algorithme du tatouage doit insérer la marque en modifiant uniquement les résultats des requêtes.

3.4 Distorsion

Un document tatoué est un document dans lequel on a introduit des modifications appelées distorsions. Pour des raisons de simplicité, le système de tatouage réalisé dans ce mémoire ne modifie que les valeurs numériques de type entier naturel. Bien sûr, il peut être étendu à d'autres types d'attributs (ex. chaîne de caractères).

Deux types de distorsions jouent un rôle très important dans l'algorithme de tatouage : les distorsions locales introduites sur chacune des valeurs d'un attribut, et les distorsions globales qui imposent une limite à l'ensemble des valeurs modifiées. Par exemple, on peut modifier chacune des valeurs d'un attribut de +/- c (distorsion locale), mais la somme de ces valeurs ne doit pas dépasser d (distorsion globale).

3.4.1 Distorsion locale

Comme son nom l'indique, c'est une modification appliquée individuellement aux valeurs d'attributs. Soit une constante $c \in \mathbb{N}$, on dit que l'algorithme de tatouage respecte la distorsion locale c, si et seulement si,
pour toute valeur modifiée, $\mid valeurmodifie - valeurinitiale \mid \leq c$.

Supposons que le propriétaire a vendu une version de sa base de données $Europe$ avec une distorsion locale c = 10 sur l'attribut $longueur$. La relation $fleuve'$, décrite ci-dessous, est une copie tatouée de la relation $fleuve$:

La table initiale $fleuve$

idf	nomf	idsource	longueur(km)	debit
83	rhone	41	880	1500
84	garonne	34	650	200
85	rhin	41	1320	2170
87	danube	49	2850	7000
89	loire	33	1022	850

La table tatouée $fleuve'$

idf	nomf	idsource	longueur(km)	debit
83	rhone	41	890	1490
84	garonne	34	655	210
85	rhin	41	1314	2170
87	danube	49	2865	7005
89	loire	33	1022	845

Ce tatouage n'a pas respecté la distorsion locale de l'attribut $longueur$, car à la quatrième ligne
$| longueur_{fleuve'} - longueur_{fleuve} | = 15 > c$

3.4.2 Distorsion globale

En introduisant des distorsions locales sur les valeurs des attributs, il faut vérifier que le résultat des requêtes ne dépasse pas une modification autorisée. Ainsi, la distorsion globale est appliquée à chacun des attributs d'une relation qui sont le résultat des requêtes concernées.

La distorsion globale est exprimée par une fonction f des valeurs d'attributs impliqués par une requête. Cette fonction peut être la somme, la moyenne, ou n'importe qu'elle autre fonction agrégat.
Soit une constante $g \in \mathbb{N}$, on dit que l'algorithme de tatouage respecte la distorsion globale g, si et seulement si, pour tout attribut résultant d'une requête,
$| f(valeur_{tatoue} - f(valeur_{initiale}) | \leq g$

Exemple :

On considère la relation $fleuve'$ précédemment tatouée, la requête suivante doit être préservée :

```
SELECT sum(debit)
FROM fleuve'
WHERE longueur <1200 ;
```

les valeurs correspondantes à l'attribut *debit* impliquées, et le résultat de cette requête avant le tatouage	les valeurs correspondantes à l'attribut *debit* impliquées, et le résultat de cette requête après le tatouage
debit 1500 200 850 **somme** **2550**	debit 1490 210 845 **somme** **2545**

La distorsion globale dans ce cas est : $| 2545 - 2550 | = 5.$

3.5 Matrice de dépendance

Les valeurs d'un attribut modifiable sont identifiées par leurs clés primaires, le nom de la table à laquelle cet attribut appartient, et le nom de cet attribut.

Soit un ensemble $\{Q_1, Q_2, ..., Q_n\}$ de requêtes à préserver avec une distorsion globale d_i correspondant à chaque Q_i , et L_i la liste des clés primaires résultats de Q_i .

Le problème consiste à trouver comment modifier les valeurs correspondant aux clés primaires k dans la liste L_i avec des distorsions locales +/-c tout en respectant la distorsion globale d_i . Dans la suite, la fonction somme sera utilisée pour la distorsion globale. Ainsi, si V_j sont les valeurs correspondant aux clés primaires dans la liste L_i pour $j \in [1, longueur(L_i)]$ alors $\mid sum(V'_j) - sum(V_j) \mid \leq d_i$.

Exemple :

marque	**-1**	**+1**	**-1**	**+1**	
clé primaire	k_1	k_2	k_3	k_4	**distorsion**
Q_1	**0**	**1**	**1**	**1**	**+1**
Q_2	**1**	**1**	**1**	**1**	**0**
Q_3	**1**	**0**	**1**	**0**	**-2**

La matrice de dépendance est une matrice dont les lignes sont les requêtes Q_i à préserver, et les colonnes sont les clés primaires k_j des valeurs des attributs modifiables. La valeur d'une cellule [Q_i, k_j] est égale à 1 si la requête Q_i a comme résultat une valeur identifiée par la clé primaire k_j, et égale à 0 sinon. En introduisant des distorsions locales (marque) égales à -1 ou +1, on obtient une distorsion globale de +1 sur la requête Q_1, 0 sur Q_2 et -2 sur Q_3. Ceci montre l'importance du choix des valeurs en ce qui concerne la distorsion locale pour avoir une distorsion globale acceptable. Dans la suite, la distorsion globale doit être nulle pour toutes les requêtes à préserver.

3.5.1 Appariement

En analysant la matrice de dépendance, on constate que la meilleure stratégie est de choisir les clés primaires, qui sont le résultat d'une même requête, par couples (p_1, p_2) et de leur affecter

une paire de distorsion locale (+1, -1) pour que la distorsion globale soit nulle. La matrice de dépendance ci-dessous montre que le choix de telles paires induit à une distorsion globale nulle sur toutes les requêtes à préserver.

marque	0	+1	0	-1	
clé primaire	k_1	k_2	k_3	k_4	**distorsion**
Q_1	0	1	1	1	0
Q_2	1	1	1	1	0
Q_3	1	0	1	0	0

En examinant cette matrice de dépendance, on remarque que les clés primaires k_2 et k_4 sont, soit présentes toutes les deux dans le résultat des requêtes (Q_1et Q_2), soit absentes toutes les deux dans le résultat d'autres requêtes (Q_3). Par conséquent, Le choix de la marque se portera sur la paire (k_2, k_4). Autrement dit, pour retrouver une paire de clé, il faut chercher dans la matrice de dépendance deux colonnes qui ont les mêmes valeurs, ainsi on est sûr que la distorsion globale est égale à 0.

3.5.2 Classification des requêtes

La matrice de dépendance citée plus haut est un exemple très réduit, en réalité elle est beaucoup plus grande car elle contient plusieurs milliers de valeurs. Afin de réduire sa taille, les requêtes sont groupées de la manière suivante : pour chaque requête Q_j, il faut trouver une requête Q_i telle que le choix des paires dans Q_i n'introduise pas une distorsion plus grande que d_j . Toutes les requêtes Q_j seront groupées dans une classe et la requête Q_i sera la représentative de cette classe. Ainsi, chaque groupe de requêtes sera dans une classe ayant sa propre représentative, et seules les classes représentatives figureront dans la matrice de dépendance.

Comment choisir les classes représentatives

Une requête Q_c est choisie comme classe représentative, aléatoirement à l'aide d'une clé secrète parmi toutes les requêtes qui doivent être préservées, pour toute autre requête Q_t , on procède comme suit :

- Calculer le nombre k de clés primaires que la requête en cours a en commun avec la requête Q_c.

- Calculer le nombre m de clés primaires présents dans Q_c et pas dans Q_t .
- Calculer la distorsion globale maximale (maxd) introduite sur Q_t par rapport à Q_c.
  $maxd = min(m, k) * max_{il}$ (max_{il} est la distorsion locale d'une valeur d'attribut à l'indice i) .

La requête Q_t est dans la même classe que Q_c si
$min(m, k) * max_{il} \leq d_t$ avec d_t est la distorsion globale de la requête Q_t .

Exemple :

clé p. val.init.att	c_1 100	c_2 400	c_3 200	c_4 300	c_5 500
Q_1	1	1	0	1	0
Q_2	1	0	1	1	0
Q_3	0	1	0	0	1

les distorsions globales sont :
(Q_1, 0.5%), (Q_2, 0%), (Q_3, 0.3%)
La distorsion locale est : 1%
$\Rightarrow$ c_1=1, c_2=4,
c_3=2, c_4=3, c_5=5.
Les distorsions globales sont :
d_1= 5% * 800 $\Rightarrow d_1 = 4$, $d_2 = 0$
d_3= 3% * 900 $\Rightarrow d_3 \simeq 3$

Ensuite, il faut calculer la distorsion locale maximale des valeurs présentes dans chaque requête. $max_1 = 4$, $max_2 = 3$, $max_3 = 5$.

Il est plus judicieux de choisir la requête Q_2 comme classe représentative puisqu'elle a une distorsion globale nulle. Le nombre de valeurs communes présentes dans Q_1 et Q_2 est m = 2, et le nombre de valeurs présentes dans Q_2 et pas dans Q_1 est k = 1, donc $maxd = min(m, k) * max_2 \Rightarrow \ maxd = 3$, maxd est la distorsion globale introduite par Q_2 sur Q_1. Comme $d_1 = 4 < maxd$, alors Q_1 est dans la même classe que Q_2.

En ce qui concerne Q_3, le nombre des valeurs communes présentes dans Q_3et Q_2 est m = 0, et le nombre des valeurs présentes dans Q_2 et pas dans Q_3 est k = 3, donc $maxd = min(m, k) * max_2 \Rightarrow$ $maxd = 0$, maxd est la distorsion globale introduite par Q_2 sur Q_3. Ainsi, Q_3 est, également, dans la même classe que Q_2. Cela signifie qu'il suffit d'introduire une marque en utilisant uniquement le résultat de la requête Q_2 tout en respectant la distorsion globale des autres requêtes.

3.6 Réserve de tatouage

3.6.1 Introduction

On appelle réserve l'ensemble des tatouages de la base de données qui respectent les contraintes d'utilisabilité. Rappelons que le but est de fabriquer un maximum de copies tatouées pour un maximum de clients. C'est pourquoi la réserve contient plusieurs combinaisons possibles de tatouages. La réserve est un ensemble de paires d'attributs modifiables représentés par le tuple *(nom_table, nom_attribut, cle_primaire)*. L'algorithme de tatouage est composé de deux étapes : l'algorithme de création de la réserve et l'algorithme d'insertion de la marque.

3.6.2 La création de la réserve

Lorsqu'un propriétaire désire tatouer une base de données avec un ensemble de contraintes, l'algorithme de création de la réserve détermine l'ensemble des paires de valeurs dont les modifications respecteront les contraintes. Ces valeurs seront stockées dans la réserve sans être modifiées. Au moment où le propriétaire voudra vendre une version tatouée de la base de données, un sous-ensemble des valeurs stockées dans la réserve sera modifié afin d'insérer une marque spécifique au client.

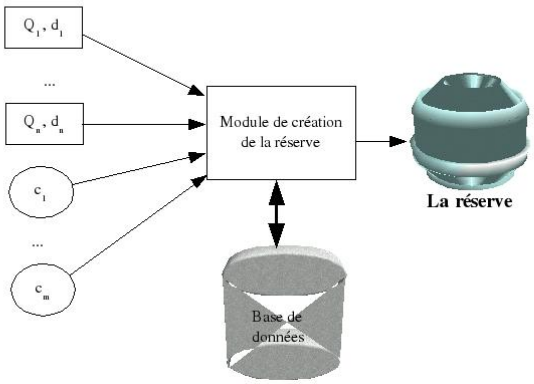


Figure 5: création de la réserve

La création d'une réserve se fait en fonction des différentes requêtes $\{Q_1, ..., Q_n\}$ à préserver avec leurs distorsions globales respectives $\{d_1,..., d_n\}$, et les distorsions locales pour les attributs de ces requêtes $\{c_1, ..., c_k\}$.

L'algorithme de création de la réserve sert à calculer toutes les séquences possibles de tatouage une bonne fois pour toute au lieu de calculer pour chaque nouveau client une séquence.

3.7 L'algorithme d'insertion de la marque

L'algorithme d'insertion de la marque s'exécute à chaque fois qu'une nouvelle version tatouée de la base de données est demandée. Il insère une marque différente pour chaque client, en utilisant une clé secrète (connue par le propriétaire) pour choisir un sous-ensemble de paires dans la réserve. Ces paires seront modifiées pour porter la marque.

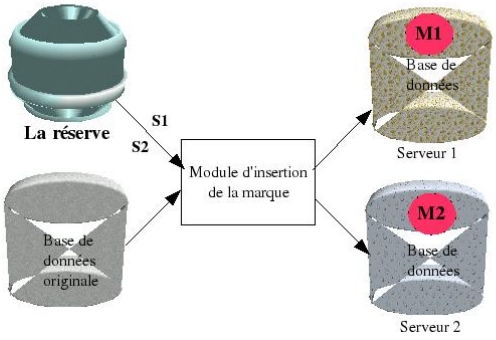


Figure 6: modèle d'insertion de la marque

La figure 6 montre que pour deux serveurs qui veulent acheter deux copies de la base de données, le module d'insertion extrait deux séries différentes (S1 et S2) de valeurs de la réserve, réalise les distorsions sur ces valeurs afin d'obtenir deux marques différentes pour chacune des copies de la base de données.

Description de l'algorithme d'insertion de la marque

Dans la réserve, il y a plusieurs paquets de paires prêts à insérer la marque tout en respectant les distorsions locales et globales. Grâce à une fonction de hachage basée sur une clé secrète, on choisit des groupes de paires (2 valeurs) dans un même paquet pour dissimuler un bit. L'algorithme pour dissimuler une marque de longueur l ayant les bits $b_1 \ldots b_l$ se déroule comme suit : pour chaque bit,

1. il choisit, en fonction d'une clé secrète, un attribut modifiable identifiable par ($table_{att}$, att_i), il détermine les valeurs des clés primaires associées aux valeurs de cet attribut dans la réserve. On considère que dans une paire de deux valeurs de clés primaires P_1 et P_2, les valeurs correspondant aux choix de cette paire sont identifiées par ($table_{att}$, att_i, P_1) et ($table_{att}$, att_i, P_2) associées respectivement aux valeurs initiales V_1et V_2. On suppose que la modification d'un attribut exprimée en pourcentage est p, la distorsion locale sur l'attribut att_i est : $d_1 = p_i * V_1$ et $d_2 = p_i * V_2$.
2. Comme les valeurs de l'attribut modifiable sont identifiées par une paire P_1et P_2, elles doivent être altérées avec des distorsions locales opposées pour que la distorsion globale soit égale à 0. Ainsi $d_1 = -d_2$, et si d_i est la distorsion locale sur les deux valeurs de l'attribut att_i, alors $d_i =\mid d_1 \mid=\mid d_2 \mid$.

3.8 L'algorithme d'identification de la marque

Lorsqu'un propriétaire est face à une base de données suspecte, il n'a pas besoin de la base originale. En effet, il construit les groupes des quatre valeurs (deux paires) de clés primaires, il calcule la marque trouvée dans la table suspecte, et la compare avec la marque donnée à chaque client. Si les deux marques ont un nombre de bits communs supérieur à un seuil donné, alors la probabilité que cette base de données a été piratée est élevée.

Chapitre 4

Contribution : Watermill et documents XML

4.1 Introduction

Le but de ce mémoire est de créer un logiciel capable de tatouer les documents XML en plus des bases de données relationnelles. Ce logiciel s'appellera par la suite Watermill, il devra apporter une meilleure adaptation entre le système de tatouage, les bases de données relationnelles, les documents XML et l'utilisateur en vue de lui faciliter le travail.

Pour tatouer les documents XML, il faut utiliser une base de données native XML pour pouvoir stocker les documents originaux et leurs copies tatouées. Ainsi Watermill pourra répondre aux besoins des utilisateurs de documents XML, et des bases de données relationnelles à travers une interface qu'il fournit.

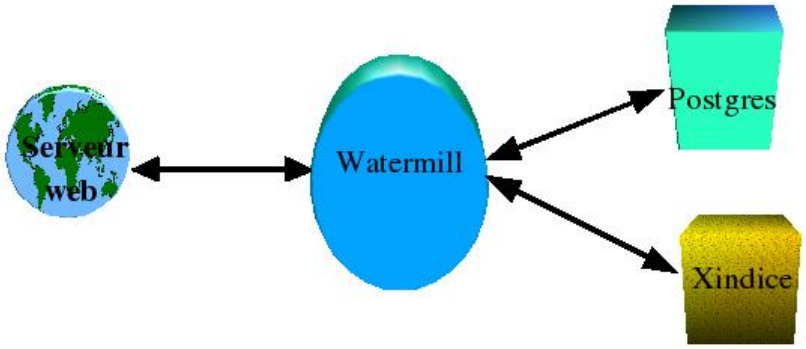


Figure 2

La figure 2 montre le mode d'interaction de Watermill avec les autres serveurs : un serveur Web qui veut acheter une donnée doit préciser si elle est de type XML ou relationnelle. Selon le type de la donnée, Watermill se connecte au gestionnaire de base de données relationnelle ou XML, tatoue une copie qu'il stocke dans la base avant de la vendre au serveur Web et conserve bien sûr toutes les informations concernant le serveur à qui il a vendu la copie de la base tatouée.

Les contributions de ce mémoire sont les suivantes :

- identifier le typage XML nécessaire pour assister le tatouage ;
- proposer et utiliser un système de stockage natif des documents XML à tatouer ;
- étendre le logiciel existant en un logiciel de tatouage générique, Watermill, permettant de travailler à la fois dans le cadre relationnel et dans le cadre XML ;
- proposer des améliorations pour l'algorithme de recherche des tatouages valides ;
- réaliser une évaluation de performances d'une partie des méthodes de Watermill.

Une partie de ces résultats a été publiée dans un Workshop international de l'ACM [14].

4.2 Schéma de documents XML

Les structures des documents XML peuvent être définies par les DTD (Document Type Definition), mais les DTD ne sont pas suffisantes pour éviter ou détecter les erreurs de typage. Si les documents doivent être fortement typés, les schémas XML (W3C XML Schéma) peuvent être utilisés.

Exemple

On considère une application d'information météorologique constituée par des stations de mesures réparties sur la France. Chaque station est équipée d'appareils automatiques relevant toutes les N minutes la température, l'hygrométrie, la nébulosité, la vitesse du vent et la pluviométrie cumulée sur les 60 dernières minutes. Ces informations sont dans un format XML qui se présente comme ainsi :

```
<?xml version="1.0"?>
<meteo xmlns :xsi="http ://www.w3.org/2001/XMLSchema-instance"
    xsi :noNamespaceSchemaLocation="/meteo.xsd">
 <obs>
    <num> 465NWC</num>
    <loc>Paris</loc>
    <date>2004-11-14T10 :30 :20</date>
    <temp unit = "celsius"> 10 </temp>
    <hygro>75</hygro>
    <nebulo>2</nebulo>
    <anemo>5</anemo>
    <pluvio>3</pluvio>
 </obs>
 <obs>
    <num> 236MYD</num>
    <loc>Marseille</loc>
    <date>2004-11-14T10 :34 :30</date>
    <temp unit = "celsius"> 12 </temp>
    <hygro>86</hygro>
    <nebulo>7</nebulo>
    <anemo>45</anemo>
    <pluvio>0</pluvio>
 </obs>
</meteo>
```

La DTD relative à ce fichier (meteo.xml) ne donnera que des règles syntaxiques que devront respecter les bulletins d'observation, mais en aucun cas, elle indiquera le typage des données et les contraintes qui sont liées à ces données. Par contre Le schéma XML rend explicites les caractéristiques d'un document XML, notamment la présence d'une clé unique pour identifier chaque donnée. La clé du fichier meteo.xml est l'attribut *num* de l'élément *obs*. Voici un extrait du fichier meteo.xsd pour décrire cette clé :

```
<?xml version="1.0"?>
<xsd :schema xmlns :xsd="http ://www.w3.org/2001/XMLSchema">
 ...
 <xsd :key name="cle_meteo">
   <xsd :selector xpath="obs"/>
   <xsd :field xpath="num"/>
 </xsd :key>
...
</xsd :schema>
```

4.3 Choix d'implantation

Les questions pertinentes qui se posent au début d'un programme concernent les outils qui seront utilisés, comme le système d'exploitation, le langage de programmation et les différents serveurs.

Le langage de programmation Java[11]

Outre les avantages déjà connus de Java, notamment son indépendance de la plate-forme (Linux, Windows, ...), la partie existante était codée en Java, il est préférable de continuer en java pour éviter tout problème de compatibilité avec le code existant.

La base de données native Xindice[12]

Le choix d'une base de donnée pour stocker les documents XML n'a pas été facile, car plusieurs produits sont présents sur le marché (ex. eXiste, tamino, Xindice, ...). Xindice[1] paraissait convenable, car c'est un logiciel libre distribué sous la licence *open source* et écrit entièrement en java. Il est livré avec le code source et les API :XML (ce qui facilite le développement), et une documentation pour expliquer l'installation et l'utilisation du logiciel.

- **Principe de fonctionnement :** les données sont stockées dans Xindice sous forme d'arborescence. Cette arborescence est constituée d'un ensemble de collections, qui ressemblent à des répertoires où l'on peut placer des fichiers XML. Cette représentation arborescente des fichiers est particulièrement intéressante, car des requêtes XPath peuvent être effectuées sur des ensembles de collections (documents).
  Comme une base de données relationnelle, Xindice permet de lire, ajouter, supprimer, et modifier des données.
- **Problèmes rencontrés :** malheureusement, peu de temps après l'installation de Xindice, on s'est confronté à des erreurs, et pas n'importe lesquelles, puisque la mise à jour des documents ne se faisait pas après une modification. Ce fut très difficile de résoudre ce problème, car il a fallu revoir tout le code source de Xindice. Finalement, deux erreurs ont été

[1] Xindice se prononce « zeen-dee-chay »

localisées, la première était à la ligne 502 du fichier XPathQueryResolver.java qui se trouve dans le répertoire « *xindice/java/src/org/apache/query* », il fallait remplacer « getOpMap()[pos+1]-1 ; » par « getOpMap(pos+1)-1 ; ». la deuxième erreur était à la ligne 105 du fichier XObjectImpl.java qui se trouve dans le répertoire « *xindice/java/src/org/apache/xindice/core/XUpdate* » il fallait remplacer « return (NodeList) _xobj.nodeset() ; » par « return (NodeList) _xobj.nodelist() ; ».

4.4 Modèle

L'objectif de ce travail est d'apporter une amélioration à l'existant pour pouvoir tatouer les documents XML en plus des bases de données relationnelles. Comme Le schéma XML décrit de façon détaillée et précise la structure de données des documents, chaque élément ou attribut modifié par le tatouage est identifiable par une clé.

Le modèle de tatouage d'un document XML se présente ainsi :

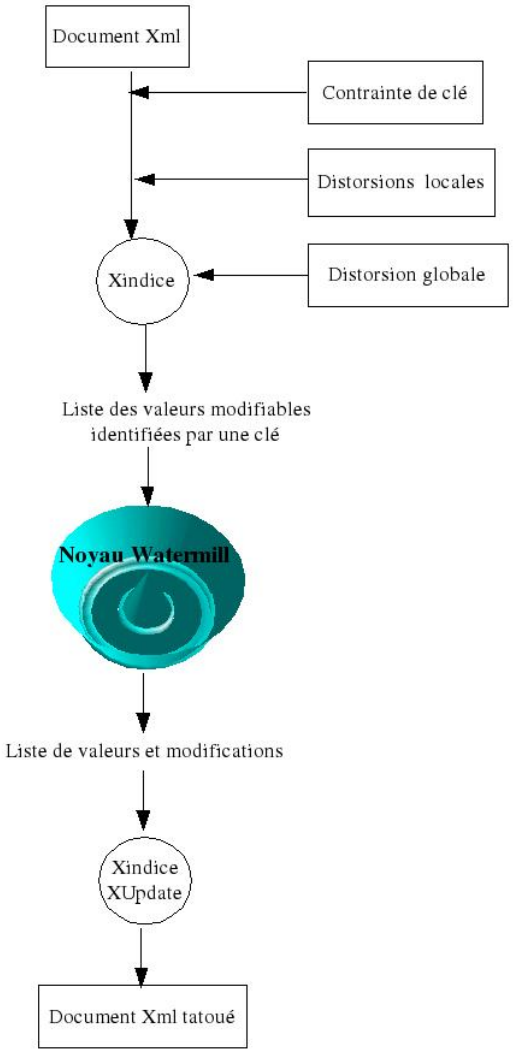


Figure 7: modèle de tatouage d'un document Xml

Revenons à l'exemple meteo.xml, tous ses éléments ou attributs sont identifiables par « *obs/num* ». Supposons que la distorsion locale est de 2 sur l'hygrométrie ayant une pluviométrie inférieure à 5, et la distorsion globale est égale à 0 sur la somme les valeurs de l'élément hygrométrie données en réponse.

Pour la distorsion locale la requête Q_1 est :

xindice xpath -c /db/meteo -q "//obs[pluvio < 5]/hygro".

L'attribut "hygro" est un attribut modifiable (à tatouer) défini par la contrainte de la distorsion locale. Il est facile de connaître les clés correspondantes aux attributs modifiables, car il suffit de transformer la requête Q_1 en une requête Q_2 qui est :

xindice xpath -c /db/meteo -q "//obs[pluvio < 5]/num".

Les attributs modifiables étant sous la forme (clé, valeur), la réponse de Xindice aux requêtes Q_1 et Q_2 est (465NWC, 75) et (236MYD, 86). On peut alors modifier l'une des valeurs de +2, et l'autre de -2, pour que la somme des valeurs avant le tatouage - la somme des valeurs après le tatouage soit nulle.

4.5 Abstraction pour les données XML et relationnelles

Il est nécessaire de modifier l'existant[13] qui consiste à tatouer uniquement les bases de données relationnelles afin qu'il puisse tatouer également les documents XML. D'autre part, les données tatouées sont identifiables par une clé unique qu'elles appartiennent à une base de données relationnelles ou à un document XML, c'est pourquoi, il faut construire une classe abstraite pour transmettre les données à tatouer.

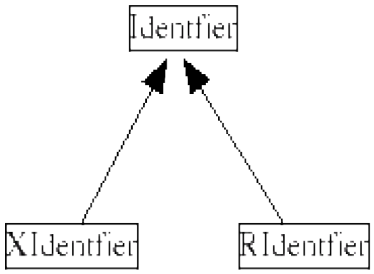


Figure 8

La figure 8 montre la relation d'héritage entre les classes *XIdentifier* et *RIdentifier* qui agissent respectivement sur les documents XML et les données relationnelles, et la classe abstraite *Identifier*.

Pour une base de données relationnelle, on utilise SQL pour obtenir ou modifier la valeur d'un élément à tatouer, identifié par son nom, le nom de sa clé primaire, le nom de la table à laquelle appartient cette clé, la valeur de sa clé primaire. Dans un document XML, on utilise des expressions XPath pour extraire l'élément à modifier, identifié par le nom de sa clé et la valeur de cette clé. Enfin, ces différentes informations sont encapsulées et transmises par la classe *Identifier*.

4.6 Proposition d'une nouvelle architecture

Watermill se compose de cinq modules principaux qui sont ***xmill*** et ***xml*** pour les fichiers XML, ***rmill*** et ***relational*** pour les bases de données et le ***noyau (kernel)*** pour le tatouage des données. Un analyseur syntaxique permet de définir si la requête concerne un document XML ou une base de données, cette requête est transmise à Watermill qui fournit une interface selon le type des données.

En supposant qu'une base de données (ex. Europe) est bien créée dans le SGBD Postgres, ou qu'une collection (ex. Météorologie) a été ajoutée dans Xindice, Watermill fonctionne de la façon suivante :

1. Dans un premier temps, une réserve contenant les attributs dans lesquels la marque peut être insérée, est créée en fonction de la donnée, et des distorsions locales et globales grâce à des commandes spécifiques.

   Exemple :
   Pour le document meteo.xml la commande est :

   create pool « nom_reserve » on « xmldb :xindice :///db/Meteorologie » with local 2 on « /meteo/obs[pluvio<5]/hygro », global 0 on query « /meteo/obs[pluvio<5]/hygro ».

2. Lorsqu'un client veut acheter une copie des données, Watermill lui attribut un identifiant et une clé publique.

Exemple :
Supposons que le client s'appelle « Marc Dubois », la commande est :

create client « marc » « Marc Dubois » « 27835 »

3. Une fois la réserve et le client sont crées, on peut demander une copie tatouée de la donnée pour le client en question.

   Exemple :
   Supposons que la copie tatouée de la collection *météorologie* pour « *Marc Dubois* » s'appelle « *meteoMarc* ». Cette copie est obtenue de la manière suivante :

   get instance *« xmldb :xindice :///db/MeteoMarc » for « marc » from « nom_reserve »*

4. Dans le cas où on est en présence d'une copie suspecte, Watermill permet d'identifier le client à qui elle a été vendue et de calculer le taux de corrélation entre la marque initiale et la marque dissimulée dans cette copie grâce à la commande suivante :

   identify "*xmldb :xindice :///*db/*MeteoMarc*"

La figure 8 (ci-dessous) donne une vision générale du fonctionnement de Watermill. Elle montre par ses différentes composantes, les interactions entre les modules Rmill et Xmill, les bases de données Postgres et Xindice, et le noyau de Watermill qui lui constitue la plaque tournante du tatouage.

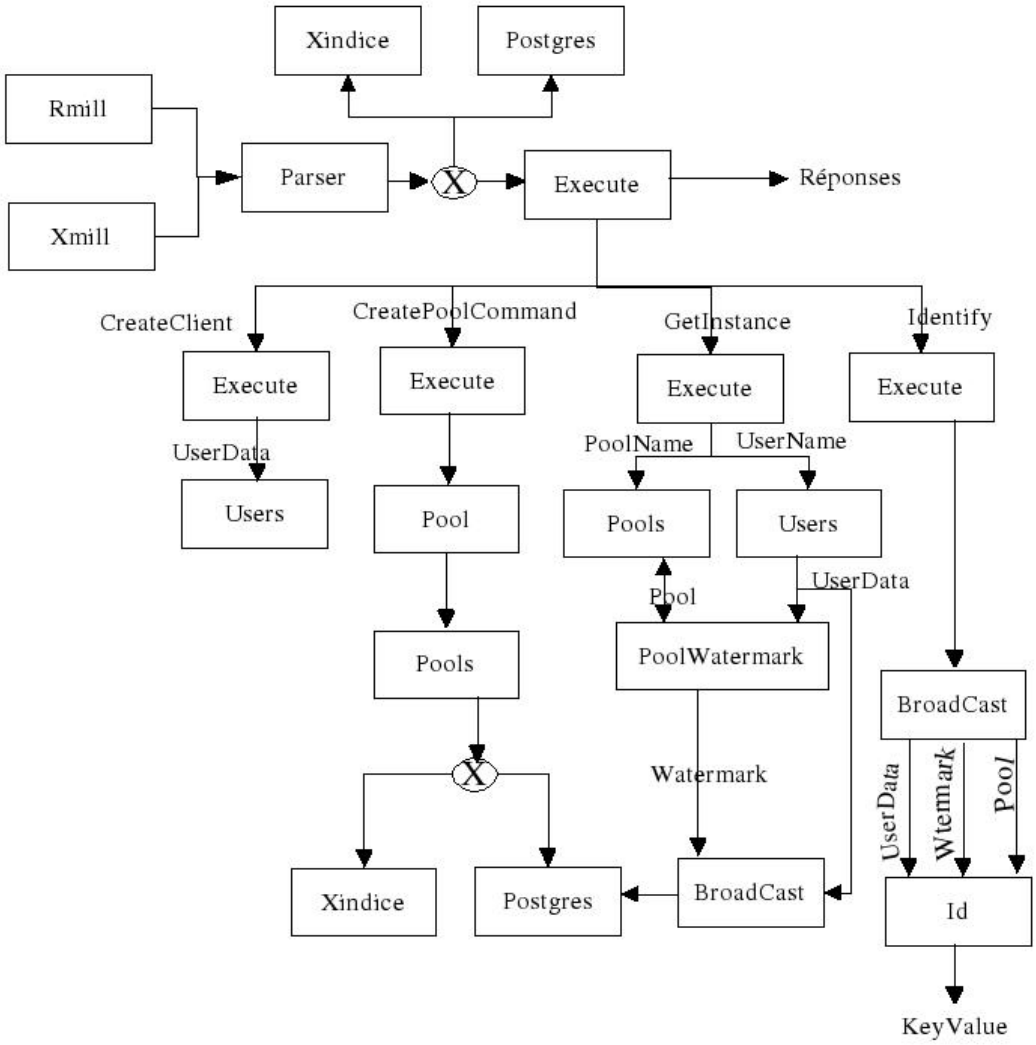


Figure 8: la nouvelle architecture de Watermill

4.7 Amélioration de la recherche des tatouages par la programmation linéaire entière

En général les contraintes d'utilisabilité sont linéaires et très liées entre elles. La programmation linéaire peut alors être une manière de trouver une solution optimale de tatouage. En effet, lorsqu'on dit que les valeurs x_i d'un attribut peuvent être altérées avec une distorsion locale n, cela signifie que les valeurs x_j de cet attribut *après le tatouage* doivent respecter les contraintes suivantes :
$x_j \leq x_i + n$
$x_j \geq x_i - n$

En supposant que d est la distorsion globale, alors les valeurs x_j doivent également respecter la contrainte suivante :

$$\sum x_i - d \leq \sum x_j \leq \sum x_i + d$$

Formulation

Tout d'abord, on construit une matrice à partir des valeurs d'attributs modifiables, cette matrice est réduite aléatoirement afin d'obtenir une série de valeurs pour chaque document à tatouer. Il s'agit alors de déterminer pour chaque valeur v_i de la matrice réduite, une modification x_i qui est dans l'intervalle [$-ld$, $+ld$] (ld est la distorsion locale sur l'attribut concerné) de façon à minimiser la distorsion globale gd appliquée à la somme des valeurs v_i.

Par conséquent, Le problème à résoudre est :

$$\left|\begin{array}{r} x_i \leq 2 * ld \\ \sum x_i \leq gd \\ \sum x_i \geq gd \\ x_i \geq 0 \\ \text{minimiser } z = \sum x_i \end{array}\right.$$

En utilisant le logiciel libre LPSolve, une solution optimale est obtenue tout en respectant les contraintes imposées. Watermill récupère cette solution et procède à la modification des valeurs d'attributs dans le document à tatouer.

En conclusion, une nouvelle option est ajoutée à Watermill qui consiste à utiliser la programmation linéaire pour tatouer les données.

Chapitre 5

Expérimentations

Contexte. Les expérimentations ont été effectuées sur un Dell Latitude D600 laptop avec un Intel Pentium M 1.7 GHz, cache de type L2 et de taille 2 MB, 512 MB de mémoire et un disque dur de 5400 RPM. Le système est un Ubuntu 4.10 GNU/Linux installation standard avec Sun JDK 1.4.2, Postrgresql 7.4.5 sans aucune mise au point spéciale. L'espace de pagination mémoire est de 1 GB.

Données de test. Les expérimentations ont été réalisées sur deux exemples différents : une base de données appelée CoverType (mesures satellite de forêts - disponible dans les archives de UCI KDD[6]), et un exemple élaboré spécialement pour le test.

- La base de données CoverType décrit plusieurs propriétés des lots forestiers, pour un totale de 581 012 tuples. Les tests ont été limités aux attributs *elevation* et *aspect*. Un attribut a été ajouté aux données pour servir de clé primaire qui n'existe pas dans la donnée originale. L'attribut aspect a été tatoué avec une distorsion locale 1. les valeurs de l'attribut elevation sont réparties aléatoirement sur 50 intervalles chevauchants. Les 50 contraintes d'utilisabilité imposent que la moyenne des aspects des données avec l'altitude (elevation) dans le même intervalle ne soient pas altérées de plus d'une unité (donc la distorsion globale est 1, ce qui est très restrictif[1]).
- L'exemple élaboré concerne une base de données relationnelles de ventes de produits dans un supermarché, avec n produits ayant chacun un prix. un certain nombre p de chariots sont remplis aléatoirement de k produits. Un tel modèle

[1] L'ensemble des contraintes est données dans notre site Web[1]

sera noté B(n, p, k). Nous avons considéré le problème de tatouage suivant pour les différentes valeurs de n, p et k : nous aimerions tatouer l'attribut coût, tout en vérifiant les contraintes suivantes :

- la distorsion sur le coût est limitée à 1 Euro ;
- la distorsion sur le total des coûts pour chaque chariot ne doit pas dépasser 1.

Notons que pour un nombre croissant de chariots, ces contraintes sont très difficiles à respecter simultanément même sur un petit ensemble de données.

Dans la suite nous avons comparé l'algorithme des paires (section 3) avec la méthode gourmande (test systématique de toutes les contraintes après insertion de chaque bit de tatouage [3]). Ces évaluation n'avaient pas encore été réalisées. Nous n'avons pas testé la version avec la programmation linéaire car le souci initial de l'équipe Vertigo était le passage à l'échelle.

Vitesse et capacité du tatouage. sur l'ensemble des données CoverType, le contrôle de 50 contraintes a pris environ 350 secondes, pour chaque bit inséré. Pour la méthode gourmande, et une marque de taille 581 bits (0.1% de la taille de l'ensemble des données), le temps de tatouage évalué est (350 * 581 =) 203350 secondes, c'est à dire plus que 2 jours de calcul. En utilisant notre méthode, non seulement le temps de calcul décroît à environ 3 heures, mais en plus plusieurs marques ont été trouvées. les résultats sont montrés dans le tableau suivant :

	méthode gourmande	notre méthode
bits dissimulés	581	143872
densité de la marque	0.1%	25%
pré-calcul (fait une fois)	pas requis	10h28min.
obtention de la première marque	2 jours	10h28min + 3h25min.
obtention d'une nouvelle marque	**2jours**	**3h25min**

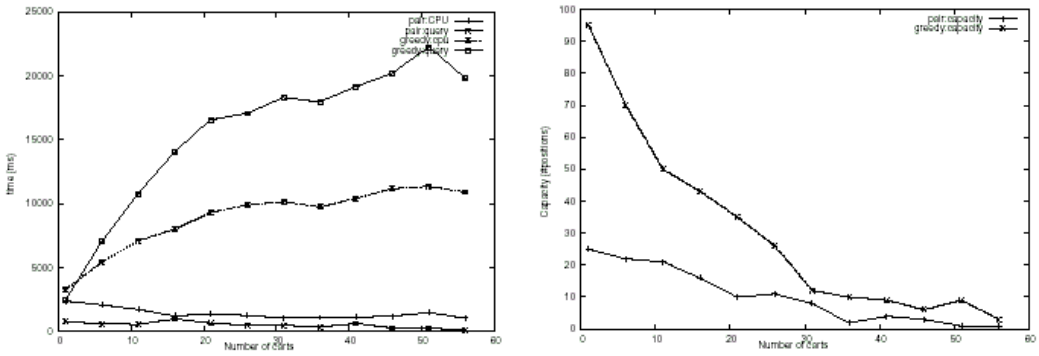
Supposons le message à insérer est « Donnée académique, client N » où N est le code binaire du nombre d'acheteurs. Ce message a 208+$\log_2(N)\,bits$. Afin d'augmenter la robustesse de l'insertion, nous répéterons chaque bit 650 fois. Le nombre de bits disponibles pour N est donc $(143872 - 208 * 650)/650 = 13$. Par conséquent, au moins 2^{13}acheteurs différents peuvent être identifiés avec cette redondance.

Le tatouage de tels documents avec la méthode gourmande demanderait la vérification de toutes les contraintes d'utilisabilité sur 581012 tuples pour chacun des 2^{13} clients. Alors qu'avec notre méthode, les contraintes d'utilisabilité sont contrôlées une seule fois sur 581012 tuples lors de la phase de compilation, et plus besoin de les revérifier pour chaque client.

Revenons à notre exemple de forêt, en une semaine, il ne serait pas possible de fournir à 4 clients des copies tatouées de CoverType en utilisant la méthode gourmande sans augmenter la capacité de calcul. Par contre notre algorithme d'appariement suffirait même sur un petit ordinateur de bureau.

En ce qui concerne la vente dans le supermarché, nous avons considéré l'exemple $B(100, p, 3)$ (100 produits, p caddies dans lesquels il y a trois produits aléatoires) pour augmenter les valeurs de p de 1 à 100. Chaque expérience a été répétée 10 fois. Le CPU et le temps d'évaluation des requêtes, la capacité du tatouage (le nombre de bits valides trouvés) et le débit de tatouage (nombre de bits tatoués par unité de temps) pour la méthode gourmande et l'algorithme d'appariement sont décrits dans la figure 9.

Premièrement, on remarque que la vitesse de tatouage est quasiment constante pour l'algorithme d'appariement, alors qu'elle est presque linéaire pour la méthode gourmande (puisque le nombre de contraintes augmente avec le nombre de caddies). Cela n'est pas étonnant car l'algorithme d'appariement n'a pas à vérifier les contraintes pendant le tatouage.



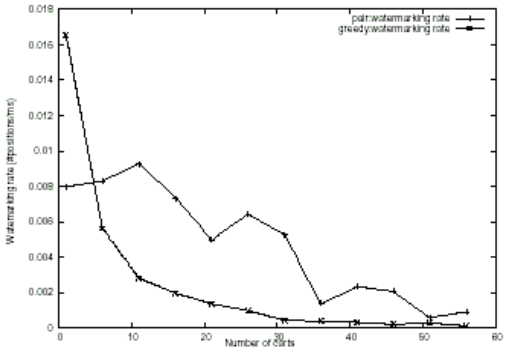


Figure 9 : Analyse temps, capacité et taux du tatouage pour la méthode gourmande et celle d'appariement.

Deuxièmement, au début (au temps 0), la capacité de tatouage de la méthode gourmande est plus grande que l'algorithme d'appariement. En effet, la méthode gourmande a la capacité d'explorer une partie plus large de l'espace de tatouage, alors que l'algorithme d'appariement est limité à un genre spécifique de solution (par exemple la recherche des paires divise immédiatement la capacité par deux). Mais les marques trouvées sont à caractère complètement différent : chaque ensemble de n positions (paires) trouvé par l'algorithme d'appariement peut coder n'importe quel message de n bits. Au contraire, la méthode gourmande trouve une marque de n position pour un mot spécifique, et il n'y a aucune garantie que les mêmes positions sont valables pour un autre.

Troisièmement, quand le nombre de caddies augmente, l'espace valide de tatouage est réduit et les deux techniques trouvent difficilement une solution. Notons que cet espace entre les deux techniques décroît au fur et à mesure.

Enfin, si nous tenons compte de la durée de calcul par rapport au taux de tatouage, l'algorithme d'appariement semble fournir des bits plus rapidement.

Chapitre 6

Conclusion

Comme nous avons pu le voir tout au long de ce mémoire, tatouer une donnée structurée n'est pas une tâche facile. En effet nous ne pouvons pas modifier un maillon dans une chaîne sans prendre en considération toute la chaîne.

Le modèle réalisé est basé sur le travail préalablement effectué par l'équipe Vertigo pour le généraliser et l'appliquer aux documents XML. Il consiste à analyser les requêtes à préserver et procède à un marquage par paires des valeurs d'attributs de sorte que la distorsion globale soit nulle. L'algorithme insère la marque uniquement dans les résultats des requêtes à préserver et se fait en deux étapes, d'abord il repère les attributs modifiables, ensuite insère le filigrane. La stratégie utilisée est l'empreinte digitale, ainsi chaque copie légalement distribuée contient une marque différente relative au client qui l'a achetée.

L'application de la programmation linéaire est proposée pour calculer les marques. Les contraintes à préserver sont exprimées sous forme d'équations et donc traduites en un programme linéaire dans un solveur (lpsove) qui résout le problème et fournit les distorsions à introduire sur les valeurs d'attributs modifiables. Cette méthode permet, non seulement, de tatouer les documents tout en respectant les distorsions locales et globales, mais en plus, la distorsion globale peut être non nulle.

Plusieurs améliorations peuvent être apportées à ce programme pour qu'il puisse être efficace et complet, on aimerait dans l'avenir :

– pouvoir préserver toutes sortes de requêtes linéaires et non linéaires.

- pouvoir tatouer toute sorte d'attributs pas seulement les attributs numériques.
- réaliser un algorithme plus robuste contre toutes les formes d'attaques.
- concevoir un programme qui analyse les requêtes à préserver et détermine automatiquement les contraintes à respecter.
- trouver une méthode pour que le système de tatouage soit aveugle (le programme réalisé n'est pas aveugle) pour ne pas être obligé de disposer du document original dans le cas d'une copie suspecte.
- passer à l'échelle la version avec la programmation linéaire.

Pour finir, des recherches sont en cours dans plusieurs laboratoires dans le monde pour arriver à une méthode de tatouage optimale appliquée aux documents structurés pour garantir leur diffusion en toute sécurité.

Bibliographie

[1] http ://cedric.cnam.fr/vertigo/tadorne/soft/soft.html

[2] R. Agrawal et J. Kiernan. Watermarking Relational Databases. In International Conference on very large databases (VLDB), 2002.

[3] R. Sion, M. Atallah et S. Prabhakar. Watermarking Relational Databases. In CERIAS TR 2002-28, 2002.

[4] Khanna et F. Zane. Watermarking maps : hiding information in stuctured data. In Symposium on Discrete Algorithms (SODA), 2000.

[5] D. Gross-Amblard. Query-preserving watermarking of relational databases and XML documents. In Symposium on Principles of Databases Systems (PODS), 2003.

[6] http ://kdd.ics.uci.edu/.

[7] R. Sion, M. Atallah et S. Prabhakar. On watermarking numerics sets. In proceedings of IWDW, 2002.

[8] R. Sion, M. Atallah et S. Prabhakar. Power : metrics for evaluation watermarking algorithms. In proceedings of IEEE ITCC02, 2002.

[9] http ://www.eurolibe.com

[10] Michelin. http ://www.michelin.com.

[11] http ://java.sun.com

[12] http ://xml.apache.org/xindice/

[13] C. Constantin. Query-preserving watermarking by constraint solving. Mémoire d'ingénieur, université polythechnique Bucarest 2003.

[14] C. Constantin, D. Gross-Amblard and M. Guerrouani. Watermill : an optimized fingerprinting tool for highly constrained data. In ACM Multimedia and Security Workshop, New York, 2005.